

NAVAL POSTGRADUATE SCHOOL

Monterey, California



19980611 023

THESIS

SECURITY ISSUES FOR THE SOFTWARE EVOLUTION MODEL

by

Anastasios X. Rambidis

March 1998

Thesis Advisor:
Thesis Co-Advisor:

Bert Lundy
Luqi

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 1998		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE SECURITY ISSUES FOR THE SOFTWARE EVOLUTION MODEL				5. FUNDING NUMBERS
6. AUTHOR(S) Rambidis, Anastasios X.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) None				10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE
13. ABSTRACT (maximum 200 words) This thesis examines the security requirements of the software evolution model and identifies possible security mechanisms called "control classes" that are applicable to the model. Then, based on combinations of "control classes," proposes a suitable security level for each of the model's databases. Furthermore this thesis deals with the possibility of using Pretty Good Privacy as a method for protection of software data stored in databases. The software evolution model captures all the necessary changes in requirements early during the development process in order to help in minimization of project cancellation, delivery delays and extra costs for fixing errors. The protection of software data against unauthorized accesses and modifications is a primary consideration for the software evolution process. In this way, we can develop a secure environment on which the software evolution can rely for accomplishing its goal.				
14. SUBJECT TERMS Database Security, Security Policies, Pretty Good Privacy, Software Evolution Model				15. NUMBER OF PAGES 116
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified		20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form
298 (Rev. 2-89)

Prescribed by ANSI Std. Z39-18

Approved for public release; distribution is unlimited

**SECURITY ISSUES FOR THE SOFTWARE EVOLUTION
MODEL**

Anastasios X. Rambidis
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, 1987

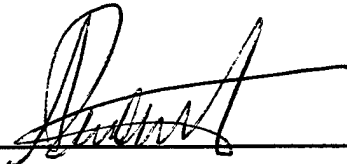
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

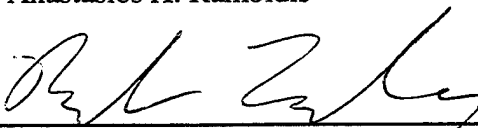
**NAVAL POSTGRADUATE SCHOOL
March 1998**

Author:

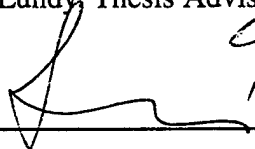


Anastasios X. Rambidis

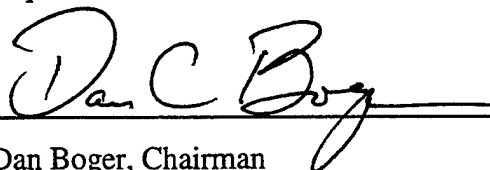
Approved by:



Bert Lundy, Thesis Advisor



Luqi, Thesis Co-Advisor



Dan Boger, Chairman
Department of Computer Science

ABSTRACT

This thesis examines the security requirements of the software evolution model and identifies possible security mechanisms called "control classes" that are applicable to the model. Then, based on combinations of "control classes," proposes a suitable security level for each of the model's databases. Furthermore this thesis deals with the possibility of using Pretty Good Privacy as a method for protection of software data stored in databases.

The software evolution model captures all the necessary changes in requirements early during the development process in order to help in minimization of project cancellation, delivery delays and extra costs for fixing errors. The protection of software data against unauthorized accesses and modifications is a primary consideration for the software evolution process. In this way, we can develop a secure environment on which the software evolution can rely for accomplishing its goal.

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	THE IMPORTANCE OF SECURITY FOR A RELIABLE SOFTWARE EVOLUTION MODEL.....	5
A.	IDENTIFYING THE WAY TOWARDS THE SOLUTION OF THE SOFTWARE DEVELOPMENT PROBLEM.....	5
B.	SOFTWARE EVOLUTION AND THE MODEL.....	7
1.	Definition	7
2.	The Need for the Software Evolution Model.....	7
3.	Description of the Model.....	9
C.	IS SECURITY A NECESSITY FOR THE MODEL?.....	17
III.	BACKGROUND ON SECURITY.....	21
A.	PROPERTIES OF A SECURE SYSTEM.....	21
B.	DATABASE SECURITY.....	22
1.	Storing the Data.....	22
2.	Access Controls.....	27
a.	Discretionary Access Controls....	27
b.	Mandatory Access Controls.....	28
3.	The Multilevel Database Concept.....	29
4.	Encrypting Data for Storage.....	31
IV.	REQUIREMENTS AND SECURITY LEVELS FOR ACCOMPLISHING A SECURE SOFTWARE EVOLUTION MODEL.	33

A.	SECURITY REQUIREMENTS FOR THE SOFTWARE EVOLUTION MODEL.....	33
1.	Organizational and Administration Requirements.....	33
a.	Protection Against Improper Access.....	34
b.	Integrity.....	34
c.	Confidentiality.....	35
d.	Availability.....	36
2.	Operational Requirements.....	36
a.	User Authentication.....	36
b.	Inference Control.....	36
B.	INTRODUCING SECURITY LEVELS.....	37
C.	CLASSIFICATION OF SECURITY LEVELS.....	41
1.	Defining Security Control Classes.....	42
2.	Definition of the Security Level.....	44
D.	IS THE HIGHEST SECURITY LEVEL THE BETTER SOLUTION FOR THE SOFTWARE EVOLUTION MODEL?..	46
V.	SECURITY POLICY AND TECHNIQUES.....	49
A.	A SUCCINCT APPROACH TO THE SECURITY PROBLEM.....	49
B.	MODEL'S SECURITY POLICY.....	49
1.	Personnel Database.....	50
a.	The Process.....	54
2.	Hypertext Database.....	57

3.	Reusable Components Database.....	63
4.	Working Revisions Database.....	69
5.	Design Database.....	70
6.	Software Base.....	76
7.	Project Management Database.....	80
VI.	ENCRYPTION USING PGP.....	85
A.	HOW PGP WORKS.....	85
B.	PGP EXPERIMENTS.....	87
C.	PGP VULNERABILITIES.....	94
VII.	CONCLUSION AND RECCOMENDATIONS.....	99
	LIST OF REFERENCES	101
	INITIAL DISTRIBUTION LIST	103

ACKNOWLEDGEMENTS

First and foremost, I must acknowledge the positive attitude and understanding I have received throughout my studies for the degree of Master of Science in Computer Science from my wonderful wife Zoi. Her endless support and patience was always a helping hand for me through the difficult phases of this work.

I also wish to express my deepest gratitude to the Professors Bert Lundy and Luqi whose scientific guidance was important for the completion of this work.

I. INTRODUCTION

Security is an important issue when dealing with software development processes such as software evolution.

The primary goal of a software evolution model is to capture all the necessary requirements for changes early during the software development process. In this way, we will succeed the minimization of:

- a. Software projects cancellation due to requirements misunderstanding
- b. Delays on the deliveries of products
- c. Total cost for fixing errors identified after the delivery of the product.

In order for this model to accomplish its goals, we need to make sure that only authorized users properly manipulate all the software data. To succeed this requirement, we have to consider establishing security mechanisms, which will provide adequate protection of data. Therefore, careful attention to security aspect is required for ensuring a well-defined security policy.

The software evolution model involves a number of different databases where software-related data is stored. For that reason, this thesis mainly concentrates on

identifying the security needs of each database and suggests some mechanisms that will provide control on the users accessing the databases and the actions that they perform on the data.

The databases could be part of a distributed system that can be accessed locally or over a network. Thus, the security of the software evolution model needs to address protection of data while it is stored and while it is transmitted over the network.

In this thesis we deal with the protection of data while it is stored in the databases. First, we define four "control classes" that are applicable for the protection of databases. Then, we define four "security levels" as combinations of the "control classes." Finally, for each database, we propose a security level based on the control classes that need to be used. Some other parameters such as data accessibility, data classification and data sensitivity are also taken into consideration.

Also, we exploited some of the features of PGP (Pretty Good Privacy) version 5.0 considering it as a method for encrypting data stored in a database. It turns out that doing encryption by record instead of by field saves a lot of space in the database.

Chapter II explains the necessity of security for the software evolution process. It also explains briefly the usefulness of software evolution and describes the software evolution model. Chapter III provides some information on security properties and presents some issues for data storage. Chapter IV lists the security requirements and contains the definitions for "control classes" and "security levels." Chapter V presents a security policy for each database of the software evolution model, suggesting an appropriate security level. Chapter VI contains some information about PGP, along with the results of the experiments we did using the PGP version 5.0 for Windows 95. Finally Chapter VII summarizes this research and presents recommendations for future work.

II. THE IMPORTANCE OF SECURITY FOR A RELIABLE SOFTWARE EVOLUTION MODEL

This chapter focuses on the importance of security for an effective software evolution model, explains the need for "software evolution" and describes the model.

A. IDENTIFYING THE WAY TOWARDS THE SOLUTION OF THE SOFTWARE DEVELOPMENT PROBLEM

Many examples in the past have shown that the complete determination and understanding of a customer's requirements is very important for delivery of a product on time. The software evolution model is a valuable tool for software engineers to avoid delays and extra cost during the development of a software product. This model allows software-related data to be retrieved from data repositories, modified according to software product needs during development and saved back to the data repositories. It is important to keep all the software data in reliable databases so that no data is lost or modified by unauthorized personnel.

Ensuring protection of these databases against inside and outside intruders is the major purpose of this thesis. Protection of data is a critical step towards the completion of a reliable model on which software

development will rely for solving software problems identified in the past.

A reliable and secure model will provide a reliable method to the software-engineering world for improving software development and maintenance efficacy, which would yield millions, perhaps billions of dollars in savings.

In the past, poorly understood requirement specifications have led to a 31% cancellation rate for software development projects according to a 1994 Standish Group survey [Ref. 1]. The same group reports that from a sample of 6,516 IT application projects only 27% of development projects were successful according to a survey conducted in 1996. 40% of the projects failed and 33% were late and over-budget.

Software-engineers encountered these software development problems because of the lack of a reliable tool that can capture and handle the requirement changes before the last phase of software life cycle.

By identifying the existing dangers to the model's security and suggesting some security solutions, we will provide the means for a steady and secure basis on which the software evolution model will be built for solving the frequent problem of software project failure.

Since the software evolution model and its security are not independent, we explain the software evolution issue and introduce a simple version of the model before we refer to the security issue of the model.

B. SOFTWARE EVOLUTION AND THE MODEL

1. Definition

Software evolution is a set of software-related activities and relations between them that affect the state of a software system. The difference with the older term "maintenance" is that software evolution refers to system's changes throughout the development of a software system while the maintenance refers to changes made after the initial development.

2. The Need for the Software Evolution Model

A look back to the past reveals many cases where inadequate capabilities for software evolution led to the failure of large projects. For example we could mention the construction of the baggage handling system for the Denver airport which cost \$20 million for requirements changes [Ref. 2] or the \$83 billion that the US invested on failed projects in FY 1995 [Ref. 1]. A well-constructed software evolution model could help to reduce the waste of money on software projects by overcoming the large backlog of

requested changes, long delays, failure to complete changes and high error rates during the development of a software project.

Currently the development of huge software projects makes it necessary to use a high level tool which could handle the complexity of real software systems. This tool needs to have advanced capabilities, so that it monitors the dependencies between activities and records how a change in an activity can affect the others, in which order and at what level. Since a change or a series of changes in the activities of complex software systems could result in the redesign of critical parts of the envisioned system, the tool should provide decision support based on the interrelations between the system's activities. The designer would then be able to identify in advance how the envisioned system will be affected by limited or extended changes due to redefined requirements.

The effectiveness of the tool is closely related to complete understanding of software evolution. For this reason the definition and implementation of a data model that simulates the difficult and complex procedures of software evolution would be very helpful.

An abstract version of this model is presented in [Ref. 3], [Ref. 4]. Salah Badr [Ref. 5] has done further development of the model. A redefined concept of the software evolution model is under research at the Naval Postgraduate School, to approach more accurately the procedures that take place during the development of complicated, real time software projects and solve known problems with previous models. From now on, this thesis refers to this redefined model.

3. Description of the Model

The software evolution model consists of a partially ordered set of steps referring to the activities that take place throughout the software development and maintenance. A graphical representation of the model is shown on Figure 1.

After the designer has constructed a prototype based on the initial customer requirements, a demonstration of the prototype allows the customer to evaluate its behavior. The customer compares demonstrated scenarios with the expected behavior of the product and identifies any problems.

The prototype's demo results in a collection of remarks, labeled "criticisms" in Figure 1. The origin of

From Prototype Demo

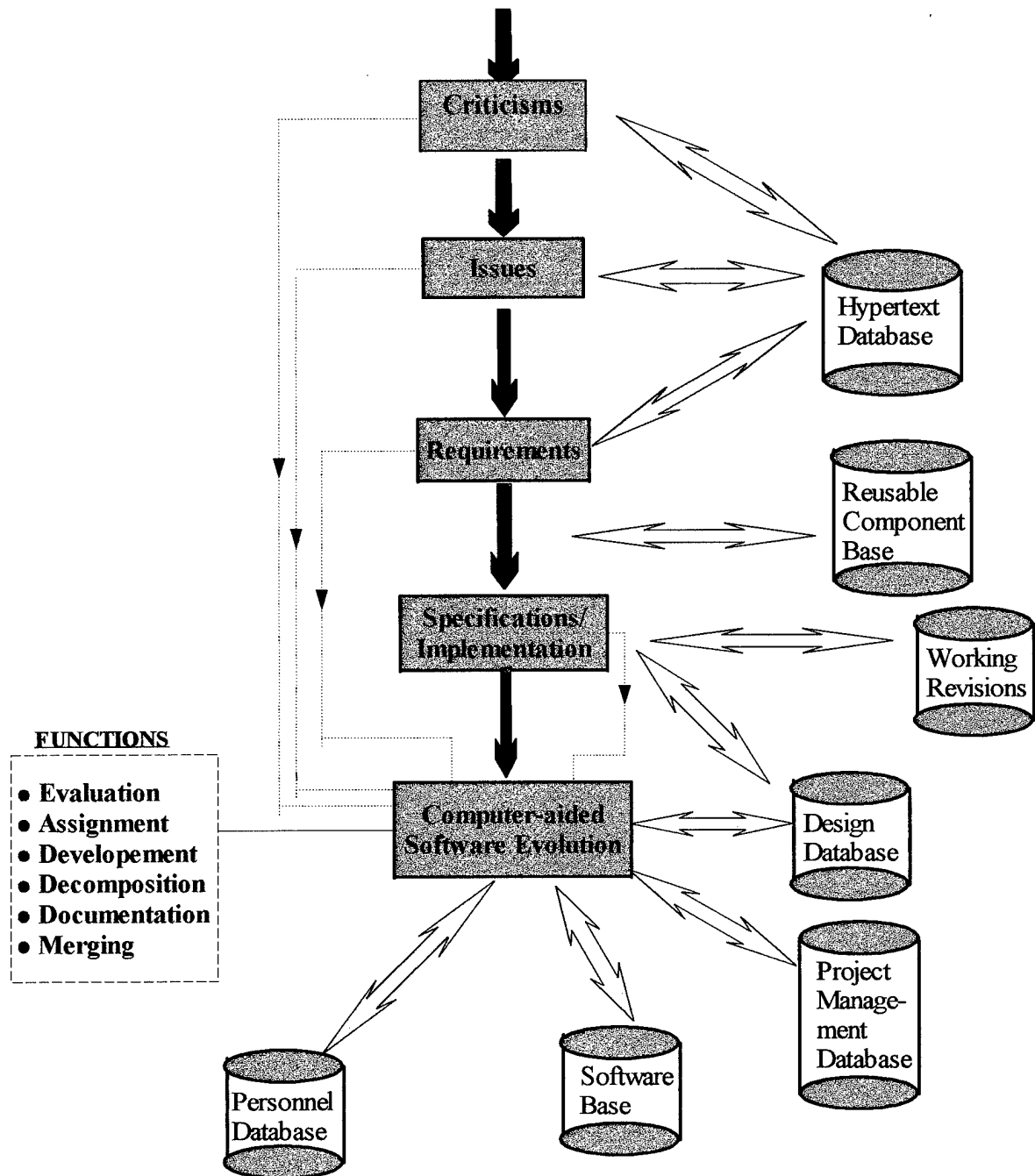


Figure 1. The Software Evolution Model

these remarks are mainly the customer(s) who ordered the product. Comments from other people affected by the proposed system are useful and must be considered, especially at the later phases and demos of the product development.

A project analysis group (usually a part of the whole software development group) under the supervision of the project manager discusses the gathered criticisms and categorizes them according to what parts of the project they address. The next action of the group is to raise some new "issues" related to each category of criticisms.

The next step is very critical for the future progress of the envisioned system toward its completion. The project group examines thoroughly the raised "issues," in order to decide if these "issues" create the need for new "requirements" or modification of already existing ones.

The project group involved in this step of the software evolution model is different than the group of people involved in the previous two steps, and is always under the co-ordination of the project manager. Project designers and analysts are people whose technical support is helpful for the evaluation of the gathered "issues." Their

participation and support in this requirements decision step is absolutely necessary.

After the requirements have been determined, the specifications for the new requirements must be created, along with the necessary modifications of specifications related to older requirements that have been changed.

The result is an updated set of specifications, which are going to affect the design of the system. These changes can be related to one or more parts of the original design plan. The size of alterations that have to be accomplished depends on how close the new requirements are to the old requirements.

After all the identified changes of specifications are categorized according to the procedures they affect, their implementation starts. This process is just before the very important last step, which is construction of the new prototype for further testing and demonstrations to the customer.

The last step of the process, "Computer-aided Software Evolution" is a complex process whose correct functionality is one of the main ingredients that determine the final acceptance of the product.

This phase cannot be considered as a stand-alone process. It is the main brain of the model and the main recipient of all the control signals coming from all the previous phases as shown in Figure 1 by the dotted lines. The outcome of the processes taken place here is to a high degree directly or indirectly dependent on those activities happening in the previous steps. So, a continuous interchange of informative signals is necessary between this phase and all the other ones. Some of the functions that take place during this vital step are evaluation, assignment, development, decomposition, documentation, and merging [Ref. 6]. The explanation of these terms is beyond the scope of this thesis.

The described model shown in Figure 1 contains six databases, which intercommunicate with various steps of the model. This is a two-way communication, which allows the users to retrieve and update the data.

There is a "hypertext database" which contains all the data related to the first three steps. All the collected criticisms, the raised issues and the derived requirements are saved mainly as text files in this database for further discussion and future reference. When data stored in this database is required, it can be retrieved via cited unique

identifiers (alphanumeric symbols) or a search can be conducted in the "hypertext database" using a keyword or a set of keywords. This search returns the results, if there are any, in categories according to the degree of relationship to the search criteria. The functionality of this database is determined by the fact that data related to the first three steps of the model needs to be manipulated often until the final version of the delivered product.

Trying to collect data from different sources and organizing them for analysis, comparison and evaluation is always more complicated and time consuming than retrieving them from a common repository. So the existence of one "hypertext database" serves the purposes of quick manipulation of hypertext data and convenient review and analysis.

Another useful database is the "reusable component database." The characteristic name of this database reveals its functionality. Components of previously developed software products are kept here ready to be used in future products. These components can be used with no changes or can be modified to meet the requirements of every new developed product. The second case is what is happening

most of the time. This is not a problem since it is not necessary for the project group to build new components from the beginning for each new project. This depends mainly on the similarity of the different requirements between the various projects. But in the case of similar requirements it saves money and time.

The "working revisions" database stores the temporal changes done to the specifications and their implementation. People working on specifications and their implementation do many changes till they finally decide how they want the specifications to look like and which is the better way to implement them. This process needs a lot of time to be completed, so there is a requirement that every day changes are kept in a database for being reworked the next day.

The "design database" is the place where the final form of specifications and their implementations are saved. Furthermore it is the place where all the documentation about the project and the source code (the actual software versions) are stored. Some of the documents it contains are the analysis report, the data flow diagram, and the module description, the testing plan and the test results.

The rest of the databases are interchanging data with the last step of the model and each of these has its own importance and functionality.

The contents of "personnel database" are not part of the design of the system itself, but the information it stores is critical for the security protection of the model. Data kept in this database is related with the people working for a project and could contain information such as Name, Address, Classification Level, Social Security Number, Tasks, and Database Access Privileges.

The "project management database" is managed exclusively by project managers and stores data that concern the projects that they lead. Schedules with deadlines for each phase of the project, schedules for tests or demos, and confidential information for personnel involved in the project might be some of the data contained in this database.

During the development of a software product it is important to have handy all the tools that might need to be used. The purpose of the "software tools database" is to provide the means that will make the designing of the product less time consuming and more efficient.

C. IS SECURITY A NECESSITY FOR THE MODEL?

The critical question that needs to be answered is the following: "do we have to consider security requirements for our model?" Since this question is very general, we should think about the security aspects of functionally vital parts of the model and whether they affect the answer to the above question. There are many definitions of "security," but the following is easy to understand.

"Security" is a set of procedures that must be followed and a set of constraints that must be met in order to assure that the data, whenever it is stored or in transit over a network, preserves its privacy and integrity. Data can be read, augmented, modified, or deleted only by those who are authorized to do so.

Having in mind the software evolution model and the definition of "security," let's try to focus on some facts about the model.

The data itself is very critical for the model to function properly. Unauthorized modification of data can lead to unexpected complications, delay to the delivery of the final product and extra cost in terms of money and manpower. So, it is evident that the protection of data is of high importance and value.

There are so many people involved in the development of a system, that it is difficult to monitor them all the time. Attempts for disclosure of confidential data by unauthorized personnel, or improper retrieval and manipulation of data from authorized personnel are activities that must be avoided using some access control mechanisms.

Another factor that should be considered is the ability of persons not involved in the project, to access critical data and retrieve it or modify it. This possibility must be examined thoroughly, so that protection mechanisms can be developed and applied to the model.

In addition to these considerations we should also consider the protection of data which is transferred over a network. We need to make sure that the data arrives at its intended destination and we need to have mechanisms that verify its authenticity and integrity.

For all these reasons mentioned so far, we are convinced that there is an imperative need to determine a minimum set of security requirements which will help in the determination of security rules. Based on these rules we should recognize those security mechanisms and techniques

that better can be applied to protect the model against harmful activities.

We will discuss some useful security issues before we proceed to the security levels and requirements of the model.

III. BACKGROUND ON SECURITY

A. PROPERTIES OF A SECURE SYSTEM

Different researchers have proposed various security properties and used their own notation and formalism. McLean in [Ref. 7] claims that properties of *confidentiality*, *integrity* and *availability* can constitute in a creation of a secure system. O'Halloran introducing *noninference* in [Ref. 8] attempts to separate the low-level activity from the high level activity. *Noninterference* introduced by Goguen and Meseguer [Ref. 9], [Ref. 10] captures the attractive notion that system security is preserved whenever high level users are prevented from influencing the behavior of low-level users. McLean in [Ref. 7] also refers to *separability* as an example of perfect security because it does not allow any interaction between high level and low level events.

Zakinthinos in [Ref. 11] talks about the *Perfect Security Property* and defines it by indicating what elements must be present in the low-level equivalent bunch for a low-level observation. He also states that a system satisfies a security property if and only if all the

low-level equivalent bunches satisfy the security property predicate P.

Building a secure system is a difficult composite task that is based on the designing of secure atomic components. Zakinthinos in [Ref. 11] identifies component independent properties that allow a designer of a secure system to interconnect components with a specific property and not be concerned about the property not holding. He also refers to component dependent properties that do not satisfy the previous statement. For these components he presents criteria that allow the system designer to know if the composition will preserve the property or not.

B. DATABASE SECURITY

Castano, Fugini, Martella, Samarati in [Ref. 12] give a simple definition for database. It is a collection of permanent data managed by the Database Management System software. According to them, databases must be reliable, protect data and programs from unauthorized modifications and disclosures and provide system continuity.

1. Storing the Data

There are two options for storing the data. One option is having different databases based on the sensitivity of the data they contain. The other option is having the same

database for all the data, no matter what the classification of data is. The bigger the number of classification the most difficult the data management. We choose two levels, "high" and "low" for simplicity.

No matter which of the two options we choose, we need to classify also the people using the databases. The classification of people should be similar to the classification of data stored in databases.

The basic architecture used in these databases is shown on Figure 2. The "trusted filter" is the most important component because it manages the data of different classifications and passes labeled data to the DBMS. Separation of data at different Security levels can be achieved using one of the following techniques.

- a. Physical. It separates spaces for data and computing resources
- b. Temporal. It separates times for use computing resources
- c. Cryptographic. It uses separate keys to access data at different levels.
- d. Logical. It uses an algorithmic separation of data within a shared resource.

Based on the basic architecture, a number of variant architectures have been developed, each one having its own advantages and disadvantages. Their differences are related

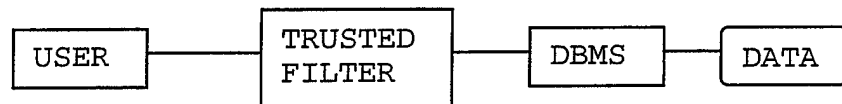


Figure 2. Basic Architecture for Databases
Classified by the Data they Contain
From Ref.[13]

with the way different classified users are accessing different classified data through different filters set-ups. Filters could be completely separated (Figure 3), interconnected (Figure 4) or fully integrated (Figure 5).

Also different classified data could be physically separated (Figure 6) or sharing the same resource but logically separated by a security kernel (Figure 7), or physically separated with synchronization (Figure 8).

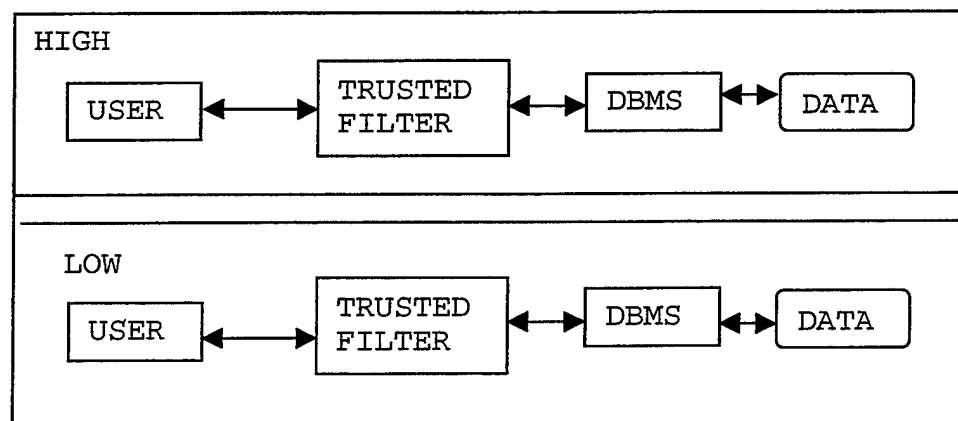


Figure 3. Database Architecture with Separate Filters From
Ref.[13]

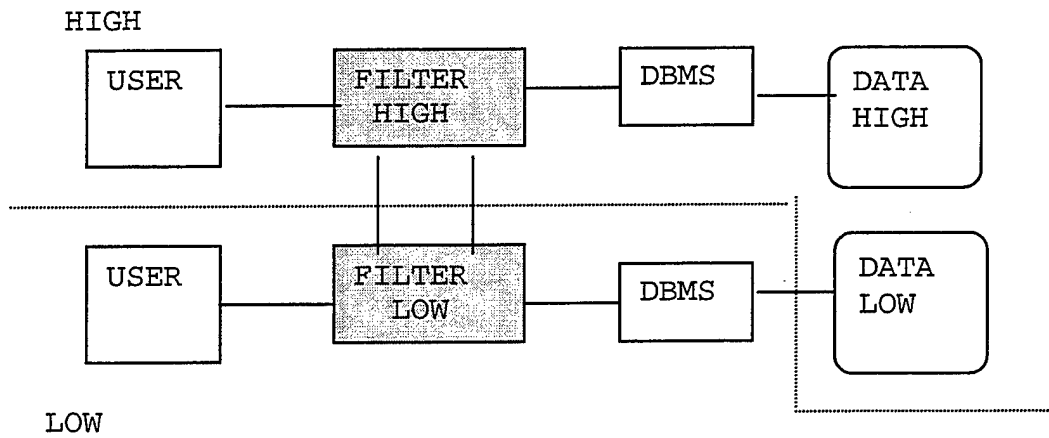


Figure 4. Database Architecture with Interconnected Filters
From Ref.[13]

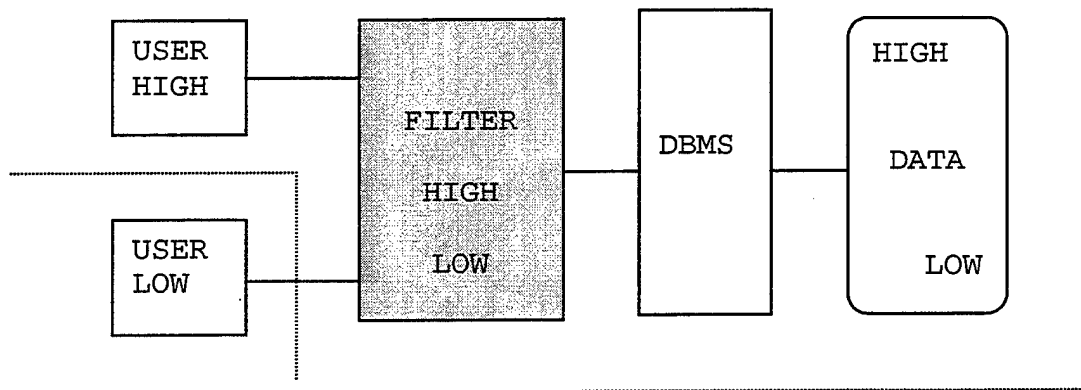


Figure 5. Database Architecture with Fully Integrated Filters From Ref.[13]

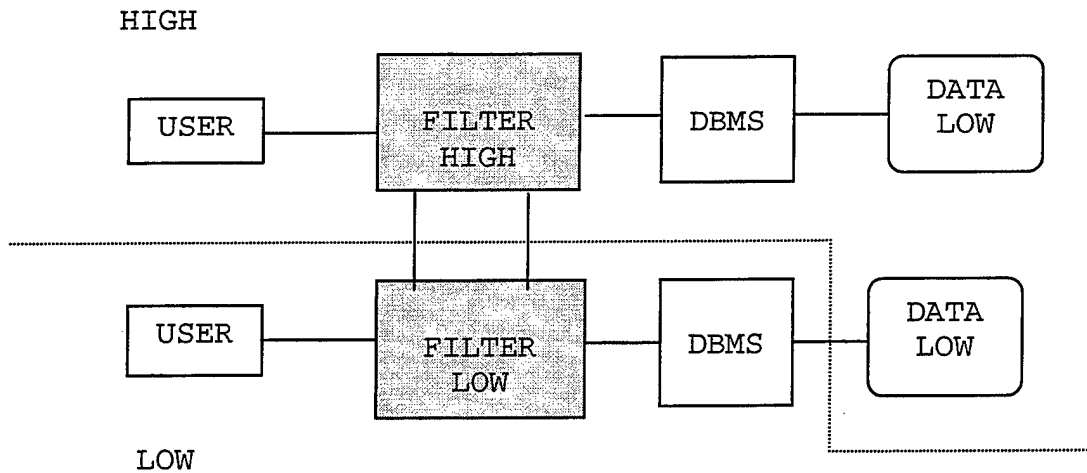


Figure 6. Database Architecture with Data Physical Separated and Communicating Filters From Ref. [13]

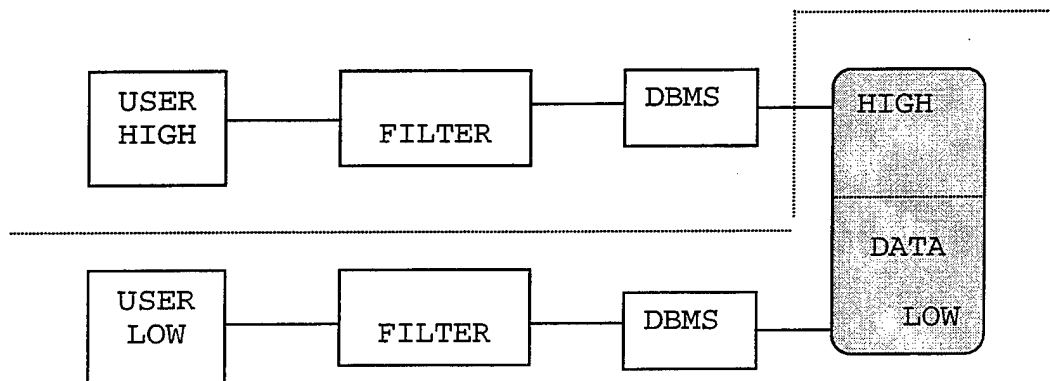


Figure 7. Database Architecture with Data Logical Separated From Ref. [13]

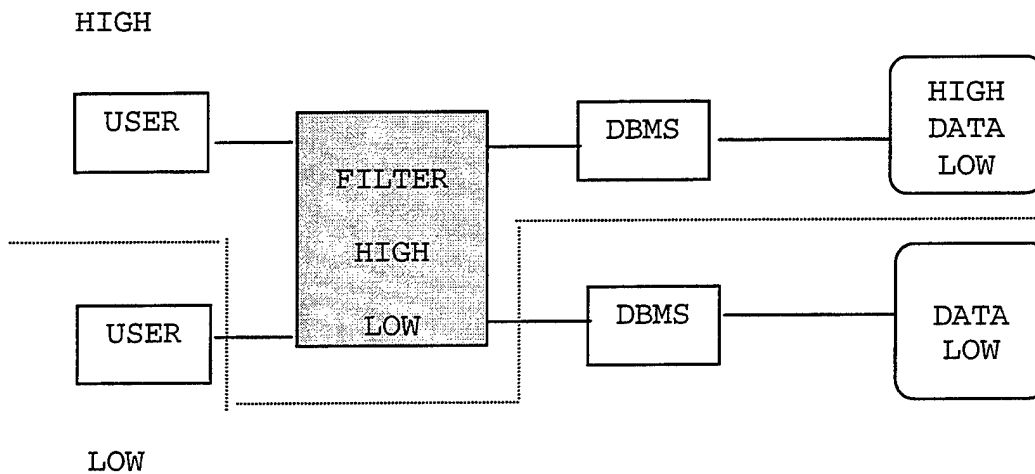


Figure 8. Database Architecture with Data Physically Separated and Synchronized From Ref.[13]

2. Access Controls

There are discretionary and mandatory access controls which according to [Ref. 14] include not only the mechanisms that are required to check whether a request issued by a particular user is allowed or not, but also all those mechanisms that are necessary to enforce the corresponding decision. Access controls are imposed by access rules, which are determined by the chosen security policies. Figure 9 is a top-level access control system.

a. Discretionary Access Controls (DAC)

Each user (subject) who creates an object is identified as the "owner" of it and usually he is the only

person that at his discretion can allow or disallow access privileges regarding this object to other users. These privileges can be changed any time.

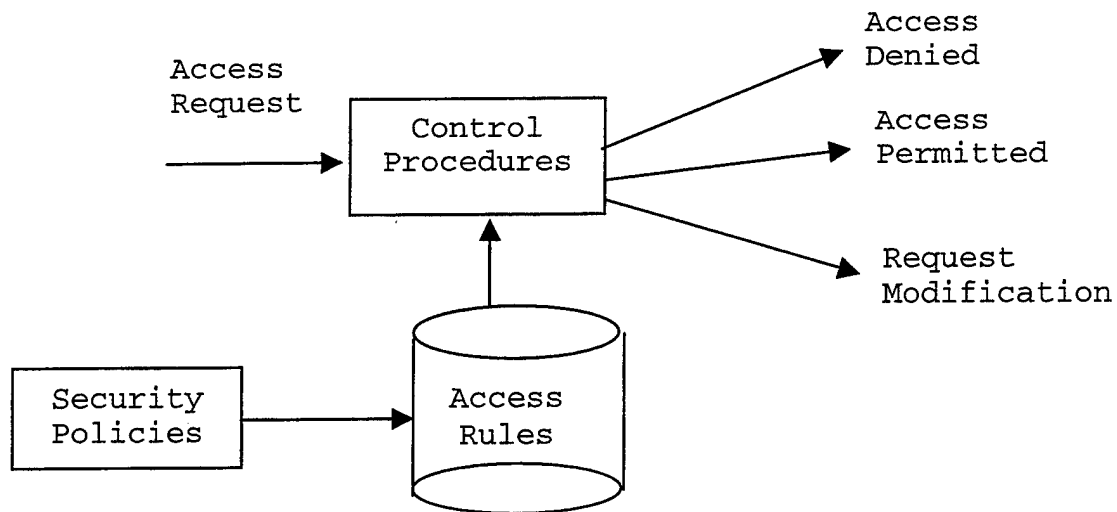


Figure 9. Access control system From Ref.[12]

b. Mandatory Access Controls (MAC)

In contrast with discretionary access controls, only the security manager (who is not the system administrator in highly trusted systems) is allowed to grant or revoke access rights.

MAC defines subject (users or programs running on behalf of users) and object security classes. An object classification defines how sensitive the information contained in object is, while a subject classification is

related with the degree of trust that can be assigned to that subject.

The combination of the two access control policies would be the better selection for a well-protected system. DAC seems to be weak in terms of losing control on privilege propagation from the owner or other authorized persons. MAC with subject and object security classes overcomes this weakness and prevents information flow towards objects of lower classification. Combining the two policies, we succeed authorization control in addition to access control.

3. The Multilevel Database Concept

All data stored in such databases are required by mandatory policies to be classified. This requirement is accomplished by associating access classes with a relation as a whole, with individual tuples (rows) in a relation, with individual attributes (columns) in a relation, or with individual elements (attribute values) in a relation.

The notation $R (A_1, C_1, \dots, A_n, C_n, TC)$, is used in [Ref. 15] to declare a state-variant multilevel relation scheme. A_i , $i=1,2,\dots, n$, is the attribute over some domain, C_i is a classification attribute for A_i and could take any value from a predefined access class set such as {unclassified,

confidential, secret, top secret} and TC is the classification attribute of the tuple.

Entity integrity is a very critical property of relational databases. The same property must be applied on multilevel relational databases. For accomplishing entity integrity the multilevel relation must satisfy the following constraints:

- a. The attributes forming the primary key must have the same access class in any given tuple.
- b. The access class for the primary key must be dominated by the access classes of all other non-key attributes in the tuple.

The problem with the multilevel databases is that we might need to have simultaneous existence of multiple data objects with the same name, where the multiple instantiations are distinguished by their access classes. This phenomenon is known as "polyinstantiation".

An attempt to try model polyinstantiated tuples and elements was introduced in [Ref. 16], where the full primary key is defined for a multilevel relational scheme. This key is equivalent to a primary key in the relational model, but now its apparent primary key, its key class, and

all classification attributes for remaining multilevel attributes distinguish each tuple.

The concept of using security classifications in a relation was the subject of many studies, which lead to the implementation of different security models. The main difference among the various models is how they deal with the problem of polyinstantiation. More details of these models can be found in [Ref. 12].

4. Encrypting Data for Storage

When a user encrypts and stores some data but then cannot decrypt it, it is impossible to go back in time and re-encrypt it. For this reason encryption applications for data storage should have some mechanisms to prevent unrecoverable errors from creeping into the ciphertext. Schneir in [Ref. 17] reports the following problems with encrypting computer data for storage:

- a. A cryptanalyst can perform a plaintext attack to the data that might also exist in a disk, or in another computer, or on paper.
- b. In database applications, pieces of data smaller than the block size of most algorithms, it is possible to cause the ciphertext to be considerably larger than the plaintext.

- c. The speed of I/O devices demands fast encryption and decryption, and will probably require encryption hardware. In some applications, special high-speed algorithms may be required.
- d. Keys are required to be safely stored, may be for long periods.
- e. Key management is much more complicated, because different people need access to different files, different portions of the same file, and so forth.

Encrypting each file with a separate key and then encrypting the keys with another key known by the users would be a solution to the problem of key management. Different users can have different subsets of the file-encryption keys encrypted with their key. This method allows multiple users to have different views of the encrypted data.

There are two options when encrypting databases. One is to encrypt the whole database and the second one is to encrypt records. The first option is problematic and inefficient since a user needs to decrypt the whole database in order to access a single record. The second option is more efficient for decryption but could be susceptible to a block-replay kind of attack.

IV. REQUIREMENTS AND SECURITY LEVELS FOR ACCOMPLISHING A SECURE SOFTWARE EVOLUTION MODEL

In this chapter we determine the security requirements for the software evolution and we define some security levels being applied to the model's databases.

A. SECURITY REQUIREMENTS FOR THE SOFTWARE EVOLUTION MODEL

We need to identify the security purposes and criteria we want in our model. Generally, we can express security requirements as constraints on states we are allowed to possess and constraints on transitions from one allowed state to the other. To define the security requirements for our model, we need to identify any constraints on the model's state. These constraints could be considered composite and expressed as the total of atomic constraints applied to each of the databases [Ref. 18].

We categorize the security requirements that need to be applied to the model in organizational and administration, and operational.

1. Organizational and Administration Requirements

Security considerations must take into account any software or hardware involved in data flow into and out of the model's databases. In addition, we need to establish a

well-defined security policy, which provides guidance on the following issues.

a. Protection Against Improper Access

Access control is the major issue for avoiding unauthorized access to data related with the software evolution model. The process of granting permission through the access control must ensure that unauthorized users are recognized and rejected while authorized users are allowed to proceed. Depending on the desired level of protection, we need to decide what access control mechanisms we will use. They could be simple or complex such as the access mechanisms applied to multilevel databases.

b. Integrity

It concerns protection of software data against unauthorized modification. It also includes protection against viruses, errors, and failures that could damage the data. Users expect to store some data in a repository, and retrieve it unchangeable unless an authorized modification has been conducted. The system should be able to inform the user if any data was corrupted.

Another aspect of integrity we need to apply to our model is related with the preservation of data consistency during concurrent transactions or the

modification of data, which is enforced to take values within an allowable range.

The issue of integrity is very important for the model since any unauthorized modification of data such as customer requirements could lead to a version of software product that is far away from customer expectation. Thus, a delay to the delivery of the product plus extra money will be required for doing the necessary corrections.

Also, the data consistency allows the people working for a software product to have the confidence that the piece of information retrieving from a repository is the latest updated data.

c. Confidentiality

This requirement prevents unauthorized disclosure and thus constitutes integrity preservation.

For the model this requirement ensures that all the information related with a recent or older version of the software product is kept within the software development group. Otherwise information flowing outside could be used from another unauthorized party which could result in unexpected situations.

d. Availability

Software data should be available any time a user requests it. Denial of service should be avoided by using mechanisms, which ensure system fault tolerance and redundancy in data, hardware and software.

Furthermore, for better protection of data related to software evolution model, the project manager should assign the task of database administrator to a trustworthy person whose duties and responsibilities should be clearly stated in advance. Also, people involved should be carefully selected and educated for security awareness.

2. Operational Requirements

a. User Authentication

People involved in the process of a software project should be identified as authorized users of the system. This authorization could be limited to some data stored in particular databases of the model. Usually authentication is succeeded through an interactive process between the user and the system.

b. Inference Control

This control is necessary for establishing a secure model. It refers to information regarding data obtained by indirect detection. A user infers information

that is not allowed to access by using some other information that he is authorized to access.

Statistical information about the software data of a project in addition to missing data (protected data not allowed to be seen) in records returned from a requested query could also constitute a dangerous inference channel.

Two other aspects for model's security requirements involve the need for auditing and data recovery. Recording all the users' activities on the data provides the flexibility of future analysis of access sequences to the software data when a compromising of data is monitored or reported. Also, it is desired to define a data recovery procedure which will be able to overcome a fatal data loss.

B. INTRODUCING SECURITY LEVELS

The different types of data stored in the databases of the software evolution model, its sensitivity along with the dangers associated with the model and the resource implications of various means of avoiding or minimizing those dangers, implies the consideration of different security levels. These are related with the number of different security steps that a user must go through for

accessing the data and the complexity of security mechanisms and access controls. The more the security steps of the process accessing the data, the higher the security level. The greater the complexity of the security technique used for database protection, the higher the security level.

The following factors, are closely related with the need for security levels:

- a. Data accessibility: how often and how many users access the data
- b. Data sensitivity: how critical for the model is the data we wish to protect or
- c. Data and users classification: data is grouped in categories based on how valuable the data is. Users are grouped in categories according to the higher classified category of data allowed to access.

We think that data accessible by a large number of people needs to be protected at a higher degree than data accessed only by a small group. The activities of small numbers of users accessing a database can be easily monitored and recorded. So, in this case we could accomplish protection of data without composite security techniques and small cost in terms of money.

In addition, if its users frequently access data, the possibility of being compromised is greater than data rarely used. So, if we know in advanced that data is accessed on a rare periodic basis, for example only once a month, then we can use a simple method for protecting it.

For better understanding of how data accessibility is related with the requirement for security levels, we will use two databases of our model as an example. On one hand, only the project managers of the developed software are allowed to access our project manager database. Therefore for this database we do need consider neither different classification levels for the data and its users nor complicated access controls. Consequently, the required security level for this database could include some discretionary access control in addition to password and login verification mechanisms.

On the other hand, "hypertext" database is accessible by a large group of people (designers, analysts, project managers) with different tasks and responsibilities. For this database it would be appropriate to establish some type of classification of data and its users so that all users can access data related only to their classification. In this way classification controllers confine the

activities of the user on the databases and we ensure better monitoring and protection of the data. People interacting with this database can manipulate only some of the data which is required to perform their tasks. The security mechanisms are more complicated and cost more money for implementation. So, in this case the security level should be higher than that applied to project manager database.

The degree of data sensitivity would be also related to the determination of different security levels. The "reusable components" or the "design" databases contain very critical information for the whole process of software evolution. Any lost or unauthorized modification of data could be disastrous for the developed system. Therefore, it would be highly recommended a well designed security method for protection of the data stored in these databases. It is obvious, that in this case the required security level will be accomplished by applying more restrictions and more controls than those for "personnel" database where loss of a record for example can not affect the progress of a project.

Of course, most of the times the realistic case we meet is a combination of the factors we are referring to in

this section. For example, data sensitivity plus frequent data accessibility could both apply to the reusable component database. A compromising of a password for accessing this database would be a disaster if the only protection were a login and password control mechanism. Certainly, we do not want this to happen. Consequently, we have to enforce highly secure methods to avoid undesired circumstances, which may be not applicable or necessary for the other databases of the model.

To conclude, different security requirements for each database along with the different characteristics of data it stores, lead us to the need of introducing security levels, which will be used as a basis for differentiating the security mechanisms applied to databases.

C. CLASSIFICATION OF SECURITY LEVELS

In order to formulate the different security requirements of databases, first we will define four "control classes." They are atomic security mechanisms that either can stand-alone or combine between them in order to establish an overall security policy. Sets of these control classes will be used for the definition of four different levels of security, each one characterized by a number. We will name these levels as security level 0, 1, 2, and 3.

The decision for the security level of each database will result from considerations related with the quantity (how many) and the identification of the proper control classes that provide adequate protection of data. Some other factors as data accessibility or data classification are also taken into consideration.

1. Defining Security Control Classes

All users requesting to access any of the databases are required to pass through a number of "control classes." The number of "control classes" is different for each database depending on its characteristics. We will try to define "control classes" under a general framework for secure databases, so that they are applicable to any database.

- a. Control class one: System requires from the user to enter a login name and password, which are checked against the information saved in the system for the particular user. In this control class, the security mechanisms of the system perform identification and authentication of a user. Security mechanisms can be improved at this control class, by the use of security add-on packages or security special-purpose

hardware, which add new security features to the system.

b. Control class two: A discretionary security model based on the DAC mechanisms, checks the access privileges that a particular user has in order to allow him to proceed to certain actions on the data. Access privileges define a user-role based security policy where the "role" declares to the system what actions should allow the user to exercise in the application. User's role is predefined in the system.

c. Control class three: Users are required in this control class to access the database via a multilevel security model. A user depending on his/her responsibility framework or in other words his/her task(s) will get just the part of information that he/she needs to perform his/her task(s). Alternatively, access to data is governed by classifications of subjects and objects in a system. Mandatory access control (MAC) mechanisms are used to enforce this security policy.

d. Control class four: Users are required to access the database via a security model, which enforces the

use of an encryption mechanism. These mechanisms could include a wide range of encryption/decryption algorithms from very simple to very complicated and highly secure. User has to decrypt those views of the data that are encrypted in order to reveal their contents. Also he is enforced by the security policy of the system to encrypt some or all of the entered data.

2. Definition of the Security Levels

We are using combination of the control classes for defining security levels. The abbreviation SL stands for Security level

- a. SL 0 is referring to systems where no protection is required. Stored data is accessible by the public and neither access control mechanism nor any security technique is required for this level. Everybody can access and manipulate the data without any restriction. None of the control classes we have defined is applicable to this level.
- b. SL 1 provides minimum protection to a system. Only control class one is applicable to this level, which forces a user to go through identification and authentication procedures for retrieving the data.

c. SL 2 makes a system more difficult to be accessed by unauthorized outsiders or insiders. A combination of control classes one and two or one and three are applicable to this level. This level provides increased protection to a system and controls the possibility of improper access from unauthorized insiders or outsiders. Alternatively, for succeeding high protection a combination of control classes one, two and three or one, two and four could be used where for the control class four an encryption method using a small size key could be used to protect the data integrity and secrecy.

d. SL 3 is the maximum protection that can be applied to a system. A combination of control classes one, two and four (using a larger in size encryption/decryption key than that in SL 2) or one, two, three and four are applicable to this level. User identification mechanisms, trusty access controls and highly secure encryption techniques enforce all the users of the system to use their authorization properly.

Table 1 shows all the security levels with the applicable combinations of control classes for each one.

The "X" declares that the control class of its row is included in the definition of the security level of its column.

	SL 0	SL 1	SL 2	SL 2	SL 2	SL 2	SL 3	SL 3
Control class ONE		X	X	X	X	X	X	X
Control class TWO			X		X	X	X	X
Control class TREE				X	X		X	X
Control class FOUR						X		X

Table 1. Security Levels Defined by Combined Control Classes

D. IS THE HIGHEST SECURITY LEVEL THE BETTER SOLUTION FOR THE SOFTWARE EVOLUTION MODEL?

The first thing coming in someone's mind would be "what is the security level that we have to apply on the model for better protection?" Of course, the answer does not take a lot of thinking, "the highest possible level."

But the highest security level is not always feasible

or sometimes it is not necessary. First we have to consider some other parameters in order to select the right security level. These parameters are related to the cost of succeeding the desired security level, the complexity of the security technique we will use, the desired speed of accessing the data and finally the type of information we need to protect. The classification of data is one of the main factors, which determines the suitable security level for a system. If the information is critical, the required security level should be higher than the case where the data is unclassified.

In addition to the parameters mentioned before, there is another factor that might affect the choice of the security level for a system, its users. Imagine a system that is very secure, although the protected data are not highly classified. The users of this system need to proceed through a relatively big number of procedures in order to access a piece of information. Such a system is difficult to use and could not be easily acceptable by its users.

To conclude, we think that the selection of the highest security level for a system is not always the right decision. The system's preservation of efficiency, flexibility and functionality could be a good feedback to

our decision to keep or modify a selected security level
for a particular system.

V. SECURITY POLICY AND TECHNIQUES

A. A SUCCINCT APPROACH TO THE SECURITY PROBLEM

What we have to face in our model case is a collection of sensitive and insensitive data that must be manipulated by people with different responsibilities who might not all be trustworthy.

The key to the methodology of accessing while protecting vital data and information should be first to identify the users. Then we must define their rights and establish their responsibilities. Everybody should follow the security rules, without any exception [Ref. 19].

The approach we chose for satisfying the security needs of the model is not unique. Since the software evolution model consists of six different databases, the security of the model resides mainly on the security of these databases.

B. MODEL'S SECURITY POLICY

In this section we establish a security policy for each database of the software evolution model. The security policy selection is related to the decision of how many and which security *control classes* are required for the protection of each database. We will discuss each database

separately and identify its security needs. Finally, we propose an appropriate security level for each database.

1. Personnel Database

For the functionality and security of the model we need to include in each person's record the following pieces of information.

- a. **Task(s):** A person can work on more than one project and also can have different task(s) for each project. This information is needed, because as we will see later on access control to some of the databases is based on the task(s) of each individual in the project.
- b. **Skill Level:** This information, in case of a security attack to the system, might be useful for the system administrator to identify some possible suspects.
- c. **Database Access and Action Privileges:** Each individual should know in advance which database he is allowed to access for manipulation of data. Furthermore, it is desirable to specify for him a set of privileges that are related with the actions, which he can perform on the data stored in databases. These pieces of information can be used from the access control mechanisms of the databases

to grant or deny permission to a user. So, it is critical to ensure that the entered information for each individual is protected from unauthorized modification.

d. **Login name, account number, and password.** The system uses them for identifying and authorizing each user and allowing him to proceed. All the passwords must be encrypted and kept in a different file (not as a separate field in a person's record).

In addition to the above critical attributes we need to identify the potential users of this database. Project managers are mainly the people that need to manipulate the data stored in "personnel" database. Project managers should be able to read, write/modify, and execute data related with "skill level" and "task(s)." Modifications of these entries by people other than the project managers are prohibited.

For the entries to "database access" and "action privileges" fields we suggest two methods. The first allows only the security administrator to assign for each individual a "database access" and "action privileges" for each database upon approval of the project manager. Project managers only may read these entries. If there is a need

for modification of these settings, they can request it to the system administrator, the only person authorized to do modifications to the values of "database access" and "action privileges" attributes.

This method allows the system administrator to control and monitor the users accessing the databases and their transactions.

The second method allows the project manager to do any necessary modifications to the "database access" and "action privileges" entries without a previous notification to the system administrator. This option is more flexible and avoids any delays because the same person (project manager) processes any modifications resulting from changes done to the task(s) entries.

In the case that the second method is chosen, every time a change to "database access" and/or "action privileges" fields is performed, a notification message from the DBMS is forwarded to the system administrator.

Keeping the number of people allowed to access the "personnel database" small implies fewer chances for unauthorized users to disclose secret info such as "login" name, "password" or pass phrase.

For this reason, we suggest that we should not allow individuals (except project managers) access to the "personnel" database. Any change to "login" name or "password" that individual needs to do, because he suspects a compromise of them, should be through the system administrator.

Any unauthorized modifications to the values of the attributes we have discussed could lead to a series of undesirable situations causing delays to the progress of the software project. For example such modifications could result in having unqualified people do specific tasks, or assigning jobs to persons that were not supposed to deal with, or people with low skill level be assigned to do tasks, which require higher skill levels. Also, it could result in having people retrieve data from databases that are not allowed.

A security level 2, defined by access control mechanisms as well as encryption to some of the data fields of the personal records, could provide enough protection against attempts for compromising data integrity. Secrecy of data is not so critical as integrity since disclosure of information kept in this database can not constitute a

threat for the scope of software evolution model, since it is difficult to affect the progress of a software project.

The security level we propose is based on a combination of control classes two and four that a user must go through every time he wants to manipulate some data. Control class one must always be the first control class that an individual has to successfully pass through no matter what the security level is. The sequence of control classes is shown on Figure 10.

a. The Process

A user enters his password and identification name (login name) in order to pass the control class one and proceed to the control class two, which includes the access control mechanisms. He enters a query and DBMS returns the corresponding records. The user, although he has the records he needs, cannot read the values of data that are encrypted. In order to be able to read the encrypted fields he needs to successfully pass through control class four. The encryption controller checks the validity of information that is required from the user and if there is a match with the respective information saved in the system, it should reveal the values of the fields

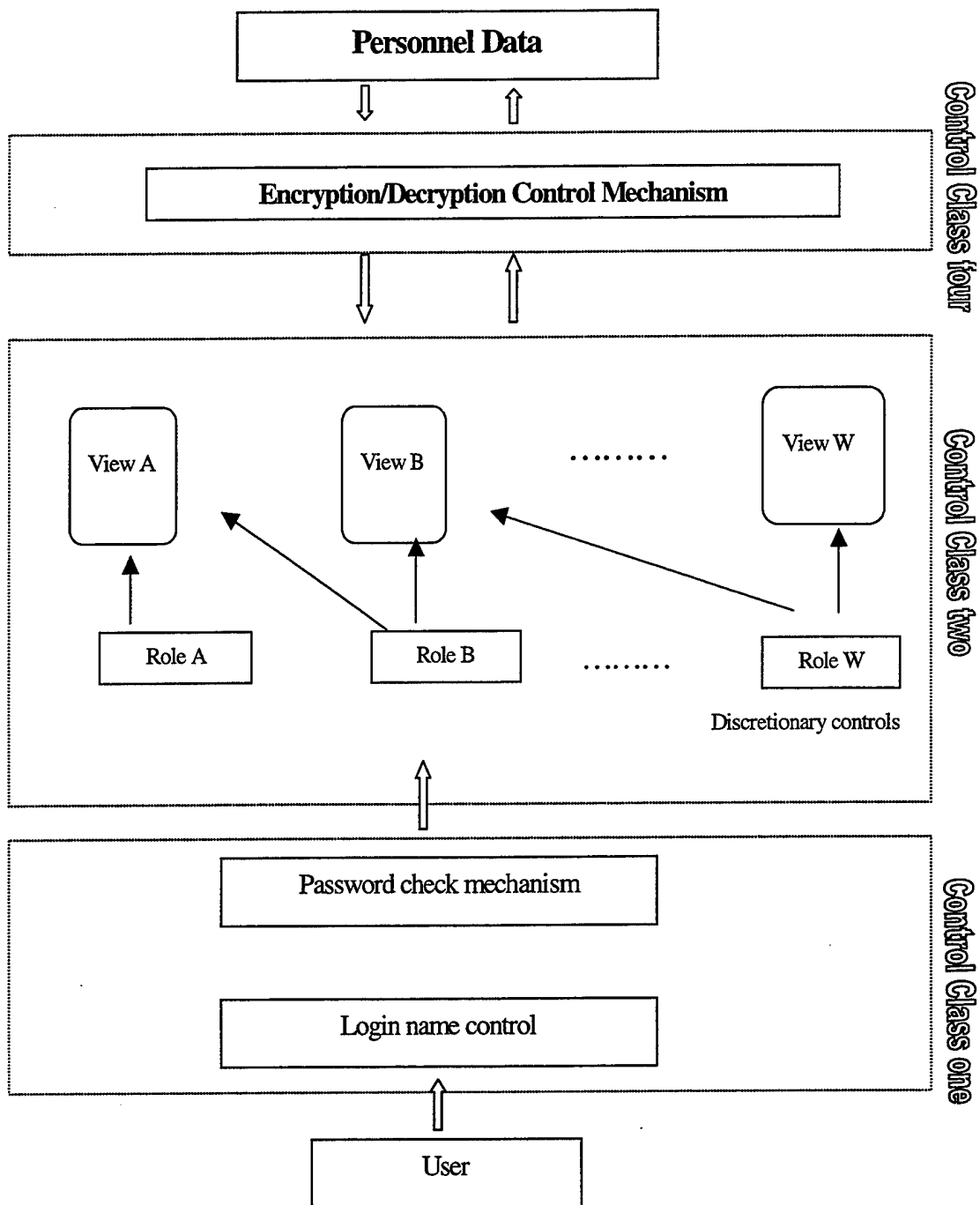


Figure 10. Sequence of Control Classes for Personnel Database

for reading. If a user needs to modify an encrypted field/record, he has to decrypt it and then re-encrypt it.

Among different encryption techniques we considered Pretty Good Privacy (PGP) for the "personnel" database. PGP is a well-known security software package combining the speed of conventional single-key encryption with the convenience and higher security of public key cryptosystems. All the users of the database should have a private key in addition to a public key. The public key is available to everybody and is saved to a public ring. The private key is kept secret and only the individual user of the database knows it.

Furthermore, the system using the PGP "sign" feature, forces the user to digitally sign any new entry or modification he did in addition to the performed encryption. This authentication scheme is needed especially for transmitting data over a network.

We did some experiments using PGP and we found out that encrypting by record rather than by field is more effective since it requires less space for data storage. Actually we found that increasing the number of fields the PGP ratio is increased, meaning that encrypting by record is even more efficient when the number of record fields is

increased. More details on our experiments are included in Chapter VI.

Users can reentry at most three times the required identification and authentication data. We believe that this number is enough to handle some misspelling or typo errors. The same number of attempts applies also to PGP control mechanism. A larger number would allow an unauthorized user to experiment a greater possibility of guessing the required entry (login name, password or pass phrase). After the third attempt, the system locks and sends a notification message to the system administrator.

A detailed graphical representation of security level 2 sequence of events is shown in Figure 11. A potential "personnel" database is shown in Figure 12 while the retrieval of data could be simulated as in Figure 13.

2. Hypertext Database

It is the place where all the criticisms, issues and requirements are stored. Any loss or unauthorized modification of them could lead to a version of the software product that does not meet the customer requirements and needs, and thus, it should be reconsidered for improvement. This reconsideration implies extra cost in terms of time and money.

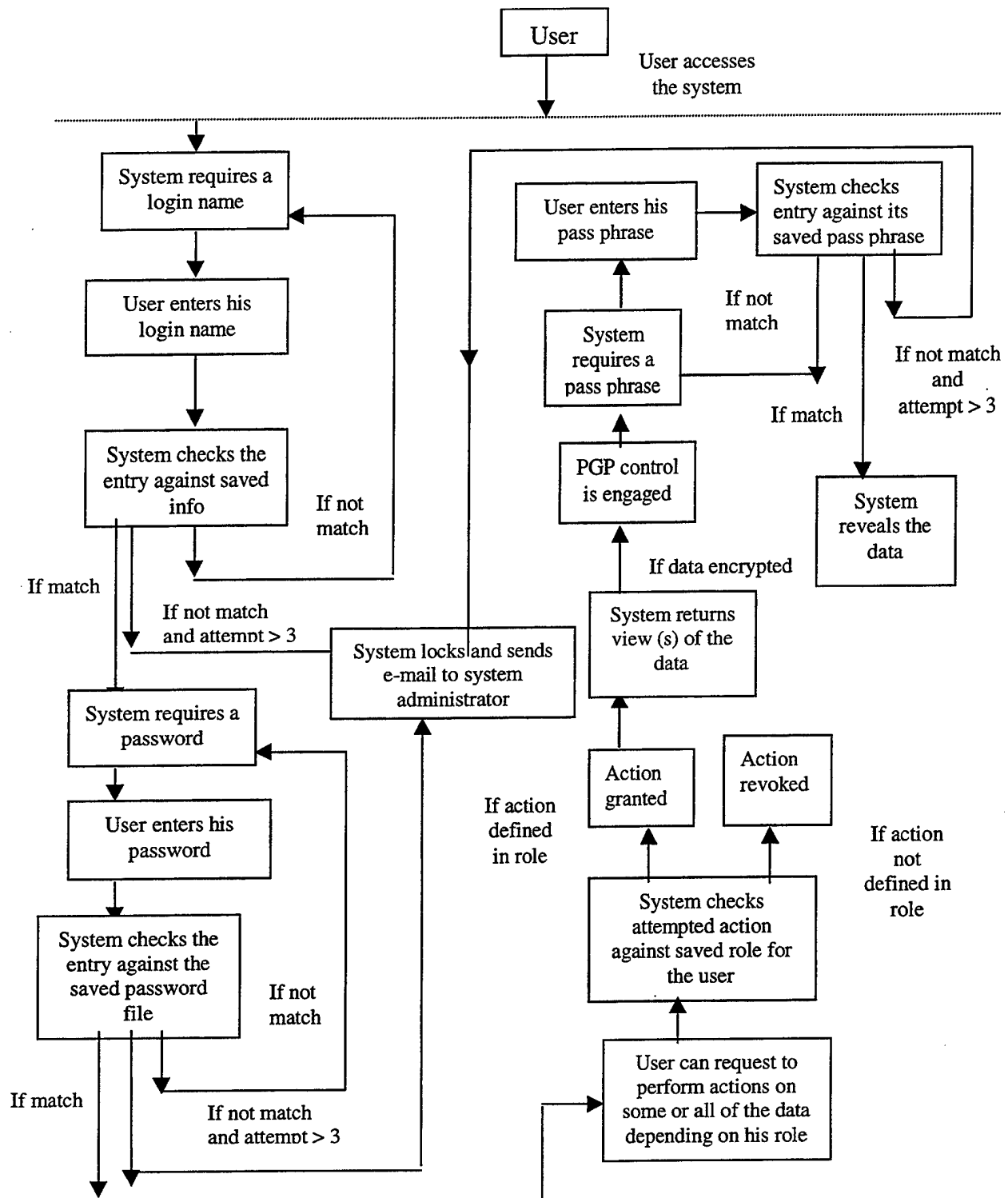


Figure 11. Graphical Representation of Security Level 2 for Personnel Database

SSN	First Name	Last Name	Account Num	Salary	.	.	.	Working on Projects	Databases Access	Classification Level
345-98-5643	Joe	Glenn	P10001	25000				->	->	CF
321-96-4321	Maria	Tern	P20004	26000				->	->	SC
123-76-0987	George	Mouse	P30009	30000				->	->	TS

Tasks

SSN	Project	Task_1	Task_2	Task_3	Task_4
345-98-5643	Proj A	Criticisms			
345-98-5643	Proj B				
345-98-5643	Proj C	Issues	Requirements		
321-96-4321	Proj A				
321-96-4321	Proj B				
321-96-4321	Proj C	Criticisms	Issues	Requirements	Specifications
123-76-0987	Proj A				
123-76-0987	Proj B	Specifications			
123-76-0987	Proj C	Programming			

Note: the symbol
-> represents a
pointer to
another table

Databases Access												
Working on Projects												
SSN	Proj A	Proj B	Proj C	Database A			Database B			Database C		
345-98-5643	Yes ->	No	Yes->	R	W	E	R	W	E	R	W	E
321-96-4321	No	No	Yes->									
123-76-0987	No	Yes->	Yes->									

Figure 12. Personnel Database

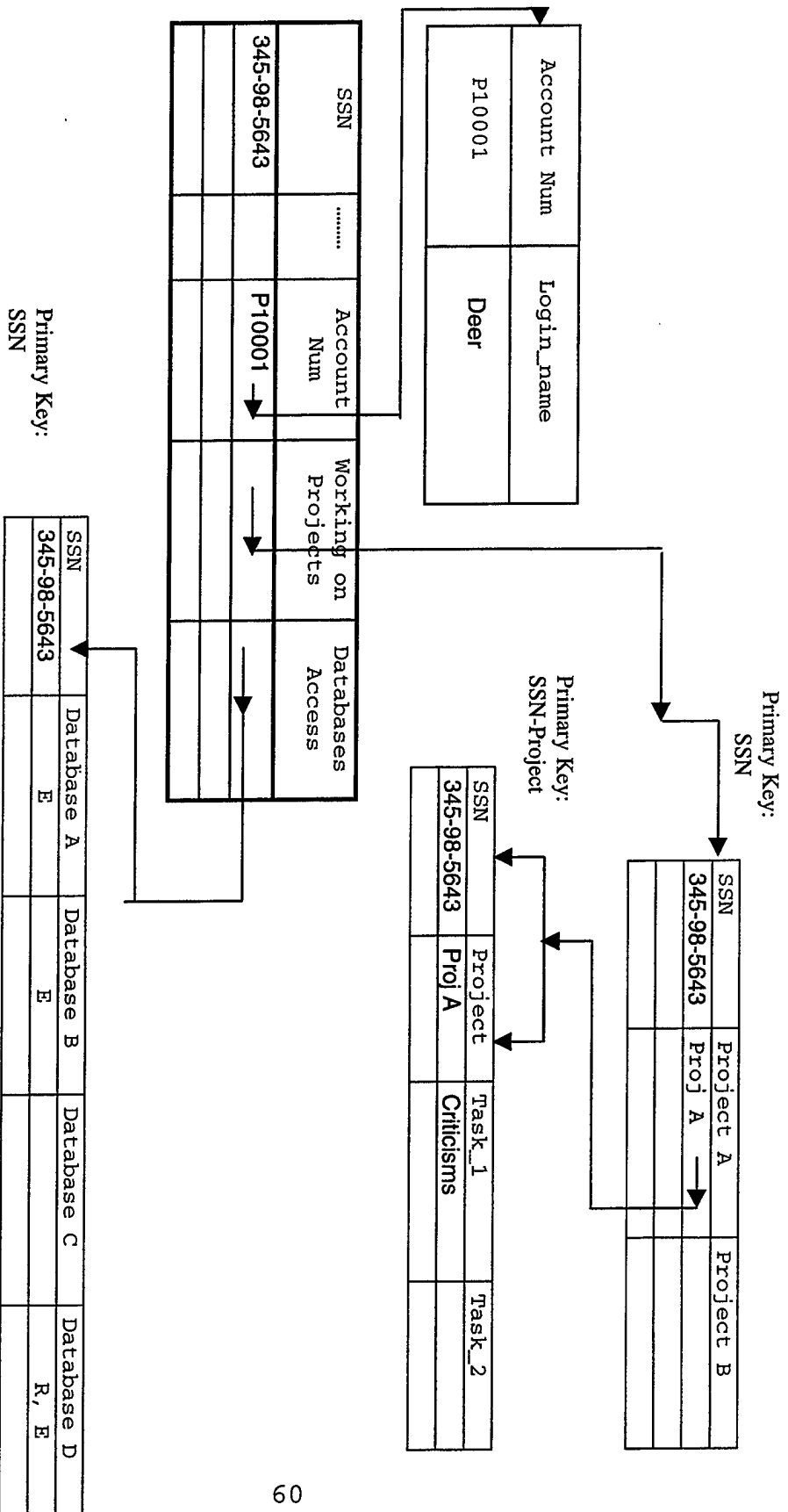


Figure 13. Retrieving Data from Personnel Database

The mechanism that controls which type of data a particular user should view, checks the login name and then tries to identify which task(s) are related with the entered login name using the task(s) information saved in "personnel" database. The system allows a user to access and retrieve only those relations, which are related with his predetermined task(s).

The actions a particular user can perform on data stored in "hypertext" database are controlled by the action privileges, which are kept in "personnel" database and set by the system administrator with the co-ordination of the project manager. "Databases access" relation of Figure 12 shows an example of how action privileges are set for the whole database. But certainly, we can set different action privileges for each of the criticisms, issues or requirements relations of the "hypertext" database. These settings must also be kept in "personnel" database.

Generally, we might have to consider that it is not necessary to allow users of "hypertext" database to make changes to the "criticisms," since we need to make sure that they keep their originality. For the "issues" and "requirements" we have to consider some modifications of the already existing ones in addition to the newly created

after the prototype demo. For this reason we need to include the "write" action privilege in "issues" and "requirements" relations of "hypertext" database.

Some other things that we have to consider for this database is the large number of people accessing the database and the frequency of access.

Since clients will be able to run a prototype demo of the software product from a remote location using a network we need to consider the possibility that criticisms will arrive to the project group very frequently, maybe every day or every hour. So updates of the "hypertext" database need to be done very regularly. Consequently, may be it is not efficient to wait for a number of criticisms to be collected and then proceed to the procedure of raising "issues" and creating "requirements."

Thus, the sequence criticisms-issues-requirements is executed in a wide range of frequency repetition depending on the project manager's policy. This means that a very high frequency of transactions with the database cannot be excluded.

A security level 2, defined by enforcing identification and authorization control in addition to multilevel access control mechanism implied by MAC security

policy ensures data integrity. Confidentiality of stored data is not of the same importance for this database since disclosure of hypertext data could not affect the software development process at the same degree with modification of some data.

The suggested security level is based on a combination of control classes one, two and three. The user is required to pass through the sequence of control classes shown on Figure 14 to access the data.

The graphical representation shown on Figure 15 presents the sequence of events taken place every time a user requests access to the hypertext data.

3. Reusable Components Database

The components stored in this database are of high importance for the software evolution procedure because they can be reused as they stand or serve as a base for building new ones. This provides great flexibility and saves time and money. It saves time because we can retrieve from the database ready to be used components in order to build and implement our specifications. It saves money because we do not need to assign people to build the components always from nothing. Doing only the necessary

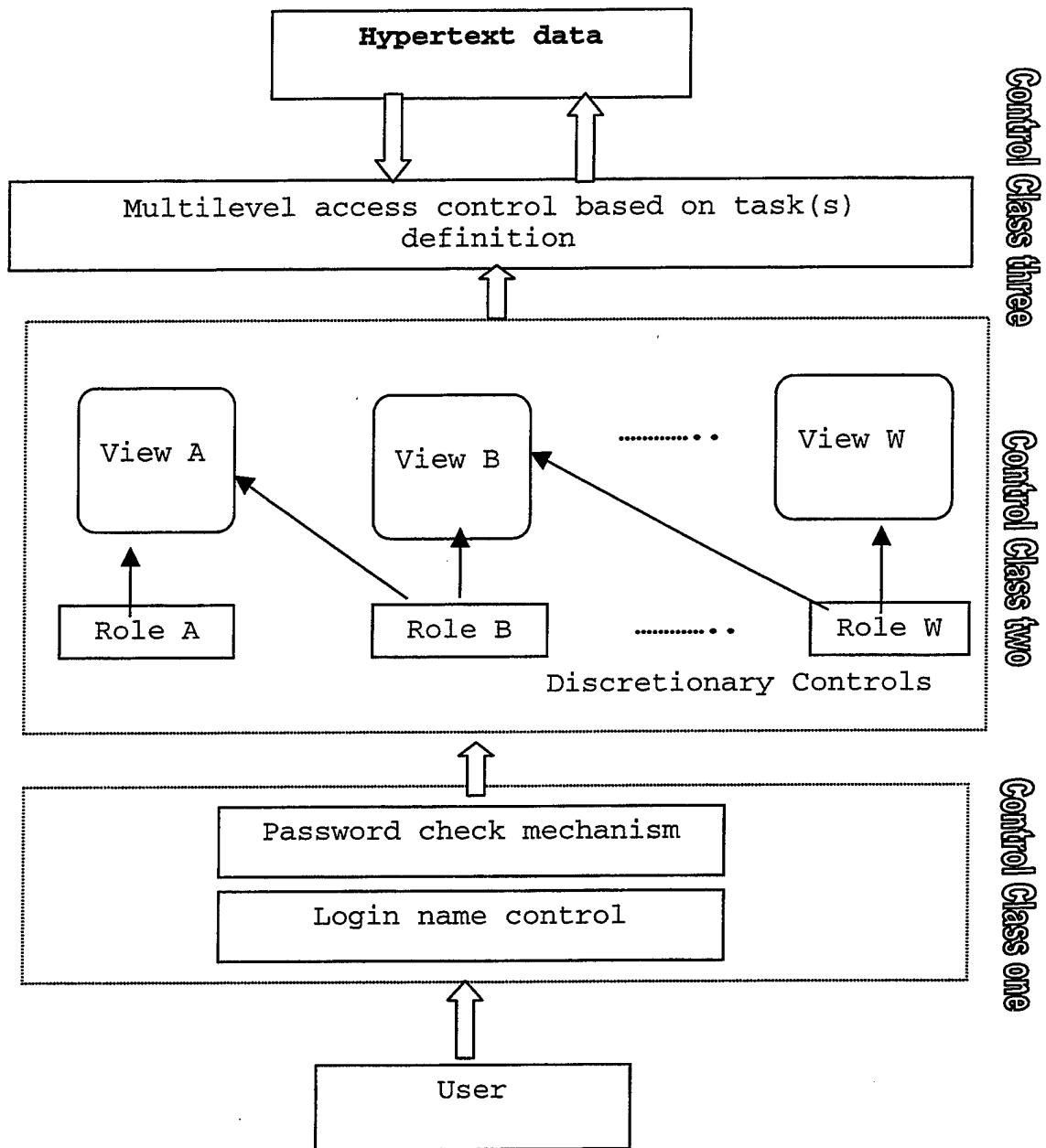


Figure 14. Sequence of Control Classes for Hypertext Database

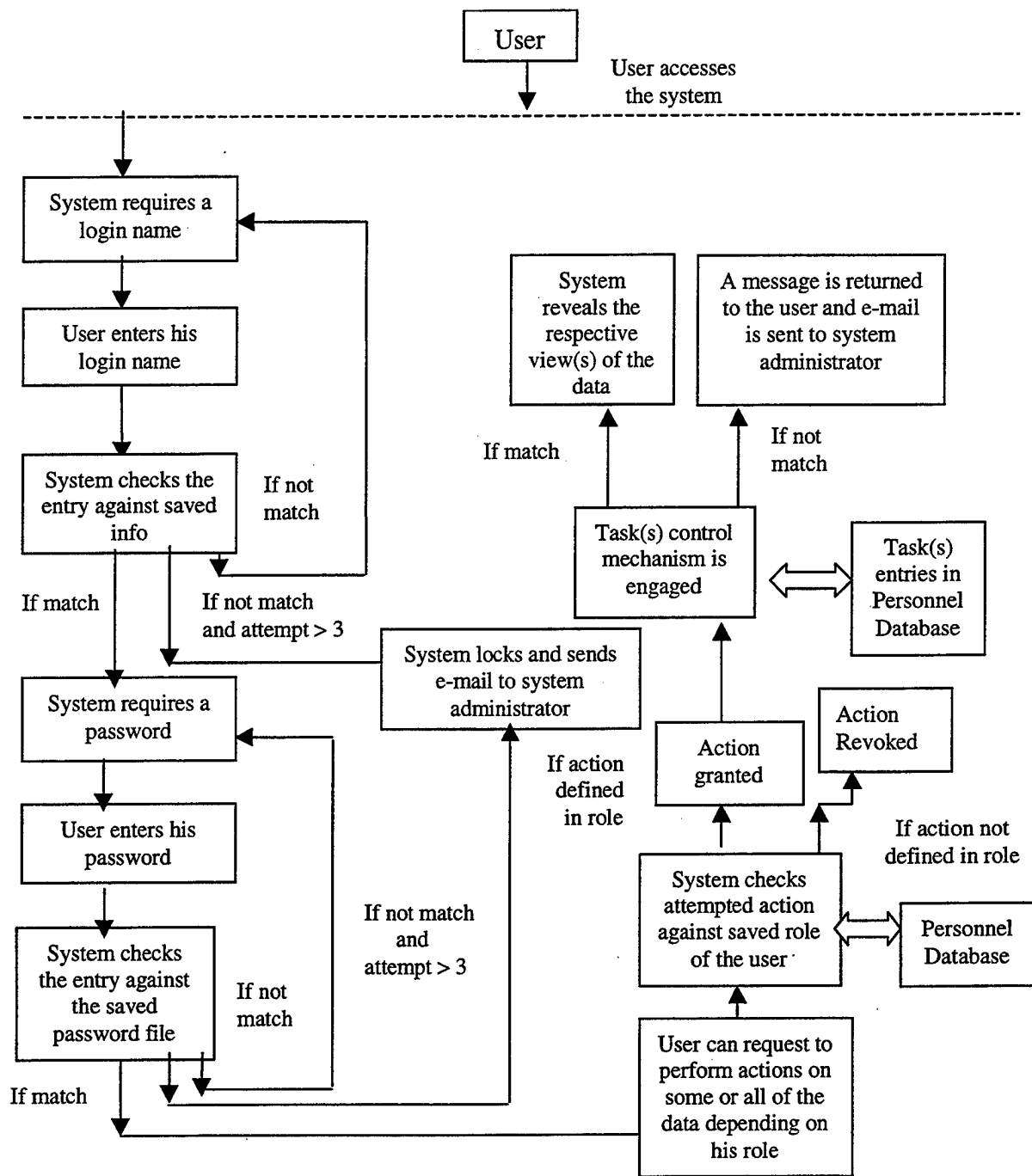


Figure 15. Graphical Representation of Security Level 2 for Hypertext Database

modifications might be enough for satisfying the current needs.

Because of the importance of these components we need to make sure that people dealing with them are trustworthy and take the necessary precautions for avoiding loss and careless or improper manipulation of these components.

Some of the components could be used for the development of highly classified systems. Thus, it is also very critical to preserve the secrecy of them.

For security purposes we need to classify the components according to the severity of damages they can cause if unauthorized people manage either to disclose or modify them. In this case, it is essential to classify also the users of the database according to the highest degree of classified data they are allowed to access.

It is very difficult to investigate when an authorized user made dishonest use of his authorization. But we can at least prevent all the users from having access to the whole data by assigning classification levels to them and the stored data.

In addition to this, having an encryption/decryption step on top of the classification control mechanism would provide extra strength to the data protection. This way,

even though someone can disguise himself as another person with higher classification level, he would not be able to view the data unless he has also compromised the secret key needed to perform the decryption of data.

A security level 3 would provide the highest protection against unauthorized transactions. This security level is based on a combination of control classes one, two, three and four. But enforcing this combination might be very complicated and difficult for the users to adapt it. Sometimes, complicated security systems are not easily accepted by people and instead of providing the highest protection, it turns out that they are vulnerable to security attacks because their users disregard difficult to implement practices.

Thus, a second option which combines control classes one, two and four might be more applicable to our model, since it effectively provides data protection without adding a lot of complexity to the system and consequently it is easier usable by the people. Either the data is stored or in transit over a network, a strong encryption/decryption method such as the PGP could ensure that the properties of secure data are met.

The problem with using this combination is that if you have no classification of data, a user having compromised the necessary key, as soon as he decrypts the data, can view it no matter what its classification.

So, users of this database should be trusted and carefully selected. Also they need to be especially careful with handling of encryption/decryption keys. For better monitoring and control of their transactions we keep an audit.

A solution that could solve some of the problems is to have the components categorized in "low" and "high" sensitivity. Having encrypted only the "high" sensitivity components and allowing only specific users to access them, we could accomplish an effective control over the activities performed on this database.

The sequence of control classes corresponding to the second option we discussed is similar with that shown on Figure 10.

For the process of accessing the data, the steps are exactly the same with that appeared on Figure 11. The important difference is the key size that encryption mechanism uses for encryption/decryption. For this database a longer key is required. For example, if we are referring

to PGP, the use of a key with 2047 bits could result in a highly protected system. The trade-off is that the process will last considerably longer.

4. Working Revisions Database

This database provides to its users the convenience of saving data to a temporal location where it is easily retrieved. It is important to ensure that the saved data remains unmodified till the next time the owner of the data decides to reuse it. This means that integrity of data is a critical issue for this database and thus we need to use security mechanisms for protecting it. Confidentiality of data is not so important at this point since the specifications and their implementation are not final.

The frequency of access to the database is high since users need to store and retrieve their work many times per day. The number of people accessing the database depends on project needs and can vary according to the task(s) assigned by the project manager.

A security level 2 defined by identification and authorization procedures along with a discretionary access control mechanism could provide adequate protection of temporarily saved data. Thus control classes one and two

are the least secure combination that we have to consider and their sequence is shown on Figure 16.

The process of accessing this database is simple and relatively quick compared to those we have seen so far to other databases. A detailed step by step access procedure is shown on Figure 17.

5. Design Database

All the data kept in this database including source code and implementation are very critical. So, the preservation of data secrecy and the protection of data integrity are the main issues for this database.

Sensitivity of data related with the various projects can vary from "very low" to "very high." This sensitivity variation creates the need for categorizing the data according to its importance. Data that its disclosure can not be dangerous at all or cannot cause undesirable situations is categorized as "unclassified." All the other data are considered "classified." For classified data, the levels often considered are Top Secret (TS), Secret (SC), and Confidential (CF). People (subjects) should also be categorized to similar with data (objects) classification levels. They are allowed to access objects that are classified up to their level.

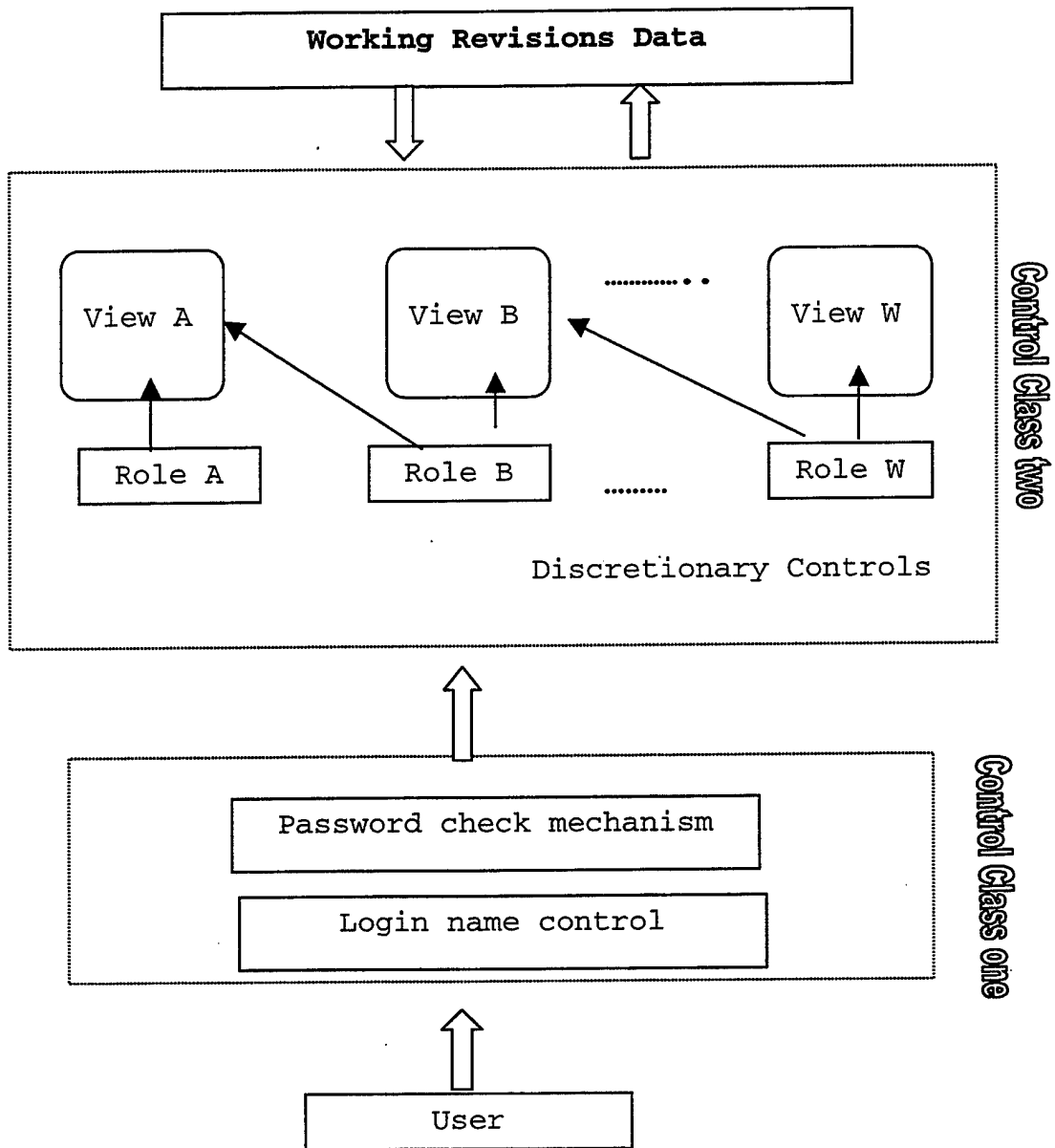


Figure 16. Sequence of Control Classes for Working Revisions Database

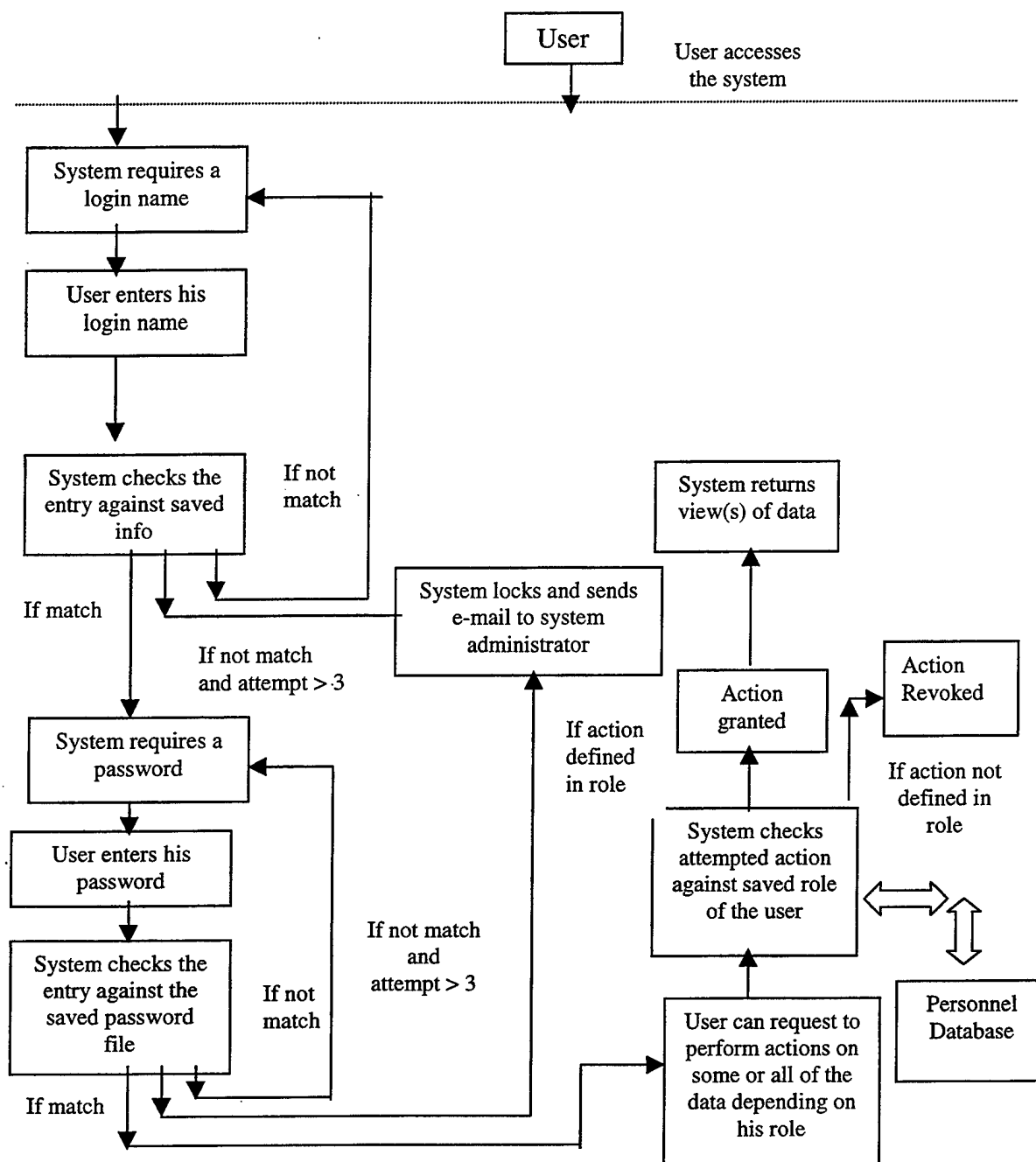


Figure 17. Graphical Representation of Security Level 2 for "Working Revisions" Database

In addition to classification levels we should use a set of compartments, as they are known. This set is not ordered as it happens to the classification levels which can be considered elements of a hierarchically ordered set where $TS > SC > CF > UN$.

The security policy, which enforces the classification of subjects and objects is based on MAC security mechanisms and uses the following principles formulated by Bell and LaPadula [Ref 21]:

- a. Only read-downward. A subject is only allowed to read objects if the access class (classification level plus set of categories) of the subject dominates their access class.
- b. Only write-upwards. A subject is allowed only to write to objects if their access class dominates the access class of the subject.

For better understanding of how the classification levels and the compartments are combined for accomplishing secure multilevel access we will use an example. Assume three clearances (classification levels): Top Secret, Confidential and Unclassified and a set of categories consisted of two elements or "compartments." The compartments are NATO and Nuclear.

Figure 18 presents a graph whose circles represent labels of "clearances" and "compartments." Arrows represent allowable transitions. The circle where the arrow points at is more secure than the circle from which the arrow starts. A transition is allowed when the circle where the arrow starts from has a label that satisfies the following:

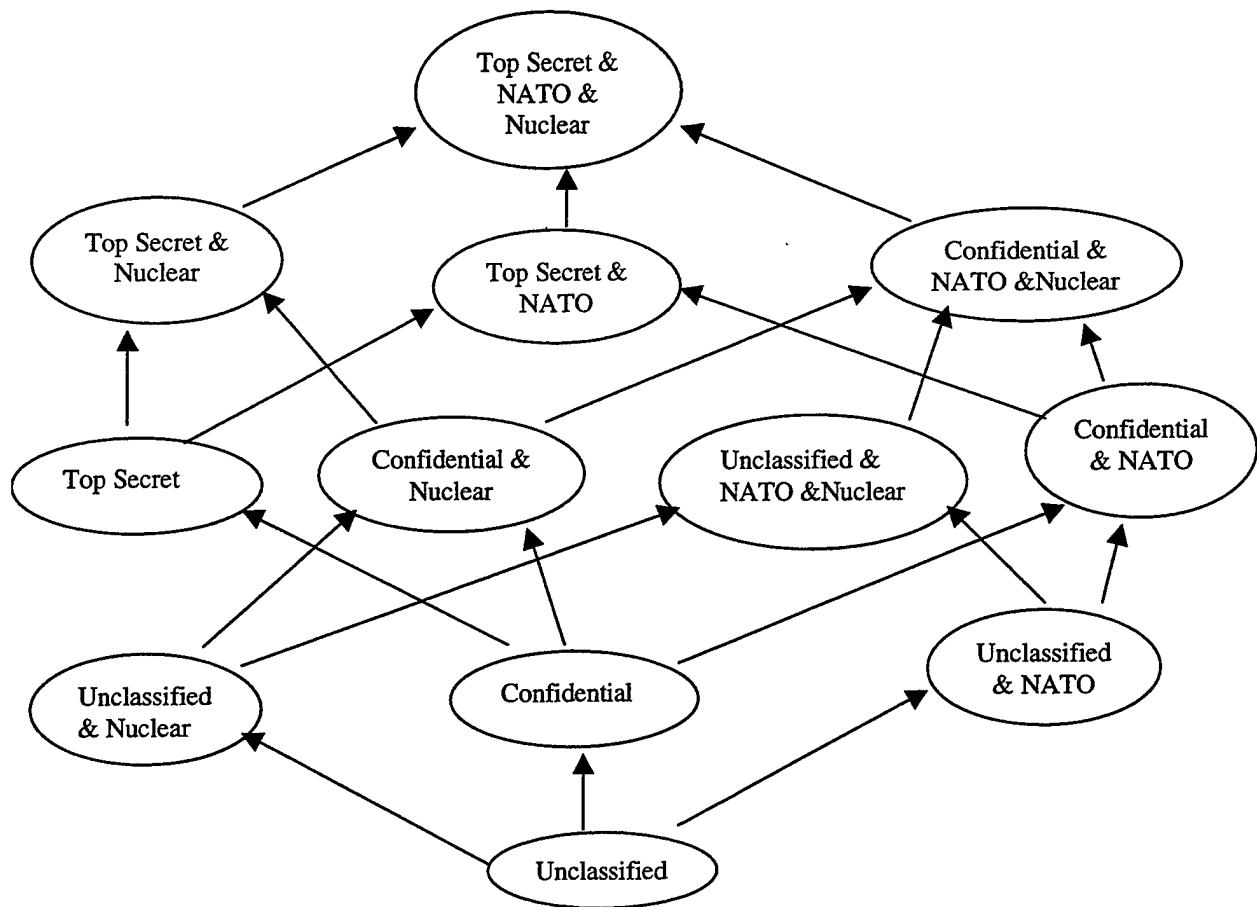


Figure 18. Allowable Flow from a Least Secure Label to a Higher Secure Label

- a. Its classification level is the same or dominated by the classification level of the circle where the arrow points at, and
- b. Its compartment is the same or a subset of the compartments of the circle that is pointed by the arrow.

A subject (person) cannot do something to an object (relation, tuple, file, application) unless an arrow or sequence of arrows points from the subject circle to object circle in the graph or unless the circles have same labels. The Security Administrator assigns clearances and compartments to each project team member according to his task(s). These settings are kept in the personnel database. They should be updated every time a person starts working on a new project or stops working on an old one. If a person is working on more than one project, which they have a different classification level, then the system administrator should assign to him the highest respective classification levels.

Since the design database stores mainly the source code and documentation for different versions of a software product, we might consider that the need for "write" action privilege should be prohibited for everybody.

The number of people accessing the database is limited and includes mainly the project managers and some designers. The frequency of accessing the data is probably low since most of the time this data is needed when a demo is prepared or when a new version of a product is ready and needs to be stored.

A combination of control classes one, two and three is probably the better selection that satisfies the security needs of this database. As a result, the security level 2 is proposed for acquiring an adequate protection of data confidentiality and integrity.

The sequence of control classes that a user has to go through is shown on Figure 19. Figure 20 presents a detailed step by step procedure of security controls a user experiences each time he wants to retrieve or store data in the design database.

6. Software Base

This is a database frequently accessed by the project members since it contains useful tools usable throughout the development of a software project. Those tools should be available to all the people involved in various projects.

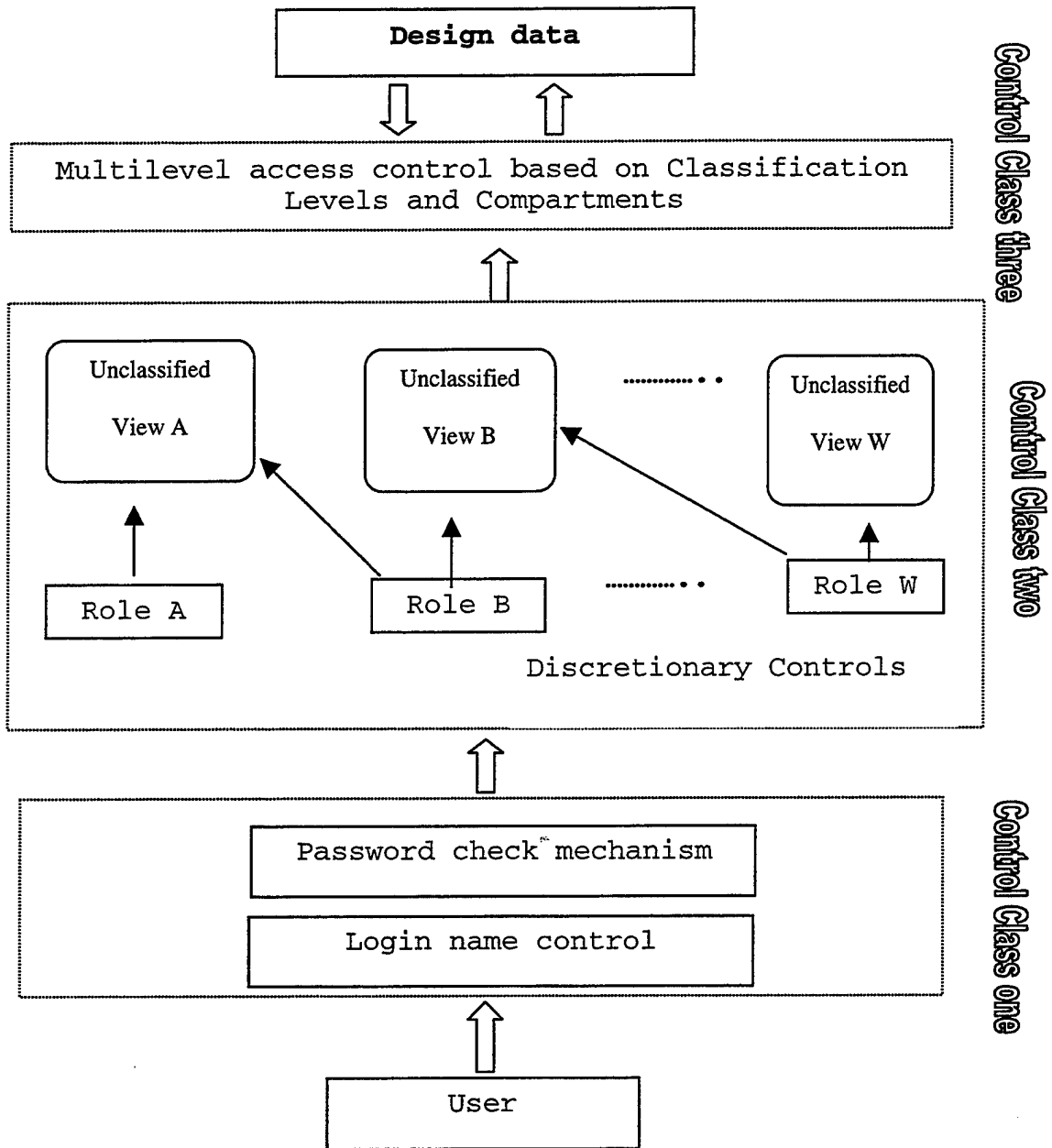


Figure 19. Sequence of Control Classes for Design Database

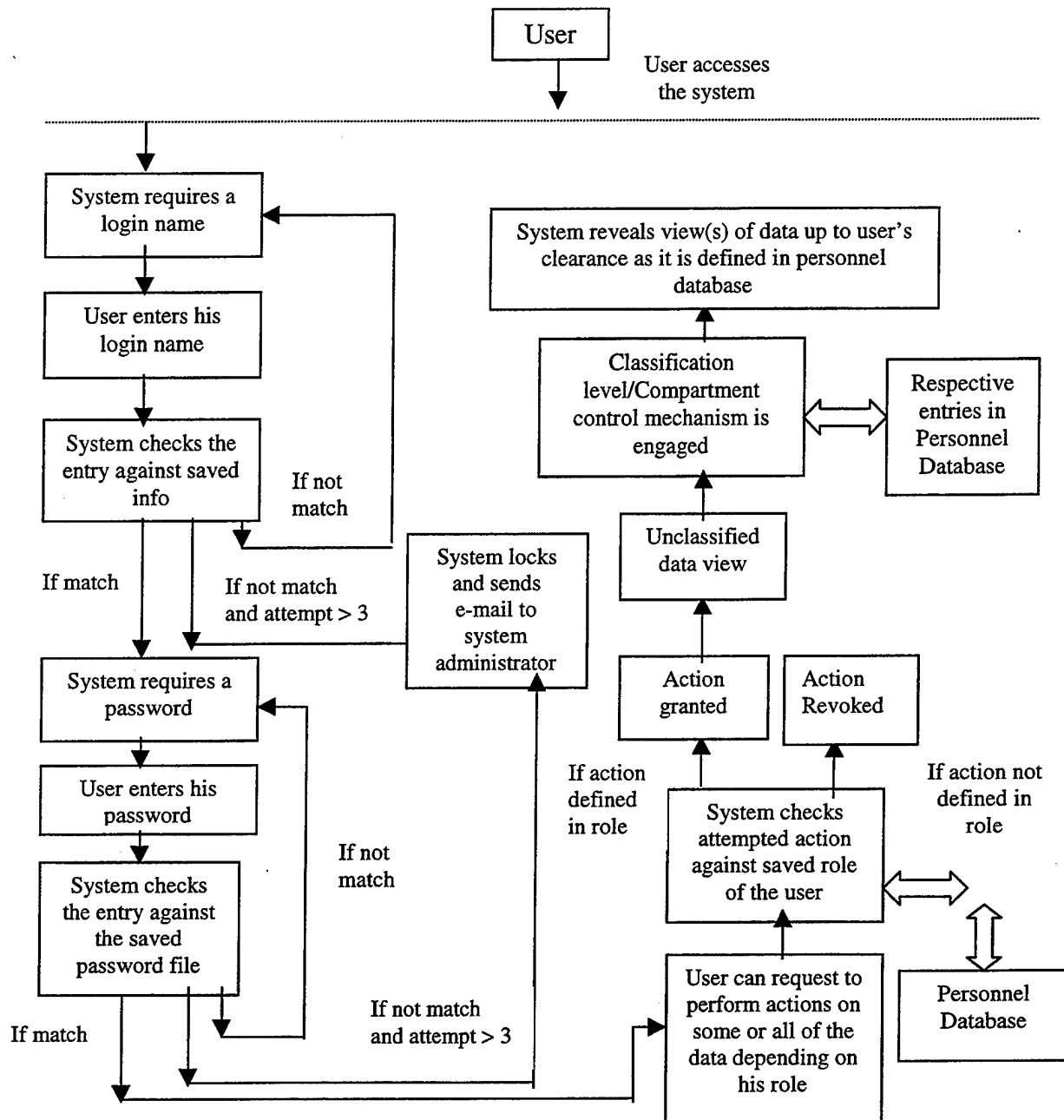


Figure 20. Graphical Representation of Security Level 2 for Design Database

An identification and authorization procedure would be enough for preventing unauthorized users to access the data. But since any try for intentional damage of data from authorized but untrustworthy users cannot be controlled only using identification/authentication mechanisms, we need to consider also some DAC limitations. For protection of data integrity, "write" action is not allowed to anyone. This does not create confinements on the process of software evolution since the stored tools themselves have no relation with newly developed data.

On the other hand, the unavailability of those tools could have a practical impact on the progress of a software project since they are necessary for some steps of the evolution process. Therefore, we have to make sure that we establish mechanisms, which are able to overcome any denial of service. We can accomplish it by using some redundancy in terms of software and hardware.

We believe that a security level 2 defined as a combination of control classes one and two is appropriate for preserving integrity and availability of tools.

The sequence of control classes and the access procedure is the same with those presented for "working revisions" database and shown on Figures 16 and 17 respectively.

7. Project Management Database

The scope of this database is to help the project managers to maintain pieces of information such as scheduled deadlines for assigned task(s), scheduled jobs, maintenance and meetings, future demos (when, where, participants) and proposed test plans.

Thus, any loss or improper modification of data is not going to affect dramatically the evolution process. Of course, if for example there is an unauthorized change in deadlines of specific task(s), that may cause some delays if the change remains unnoticeable to the project manager. Again, this is controllable and sooner or later the project manager will realize any suspicious changes.

In addition to project managers, may be some designers need also to maintain their data in project management database.

Generally the total number of people that need to have access to this database is limited. The frequency of accessing the database is high.

A data disclosure or modification cannot create serious threats for the smooth continuity of evolution process. It might cause some temporal delays but it is difficult to affect the delivery of the product. Lack of

availability should not be a problem since some temporal arrangements can give solutions to such cases.

Therefore, there is no need for data protection more than security level 1, which uses only an identification and authentication procedure (control class one). Figures 21 and 22 show the control classes and the access procedure steps respectively.

If we need to consider a higher protection of this database the highest security level that we should suggest is 2, which adds some control on the allowable actions performed by the users. In this case the sequence of control classes would be as in Figure 16 and the access procedure will be the same with that on Figure 17.

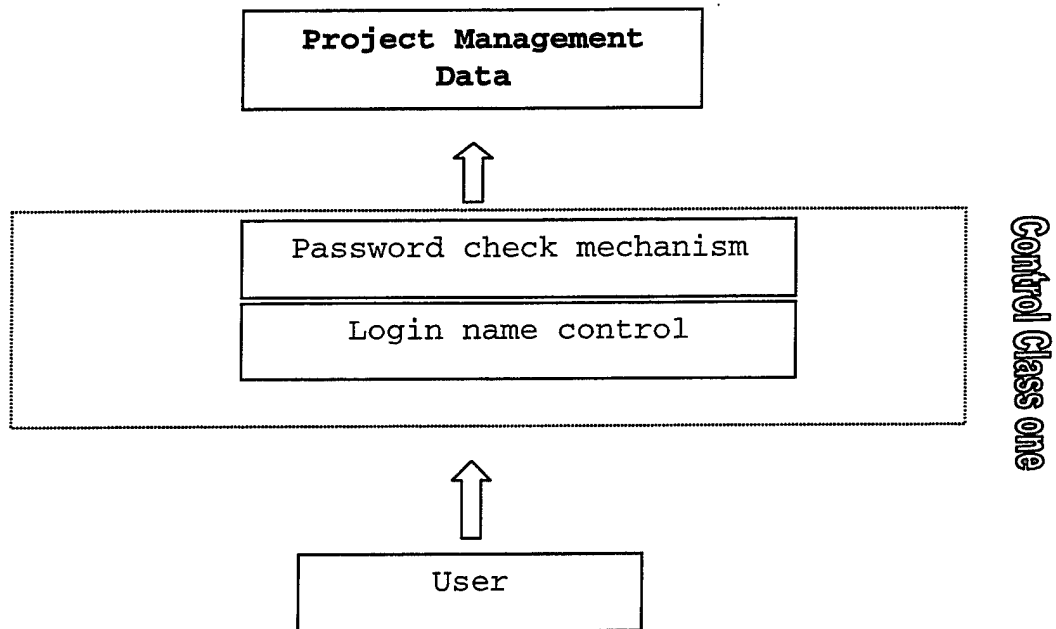


Figure 21. Sequence of Control Classes for Project Management Database

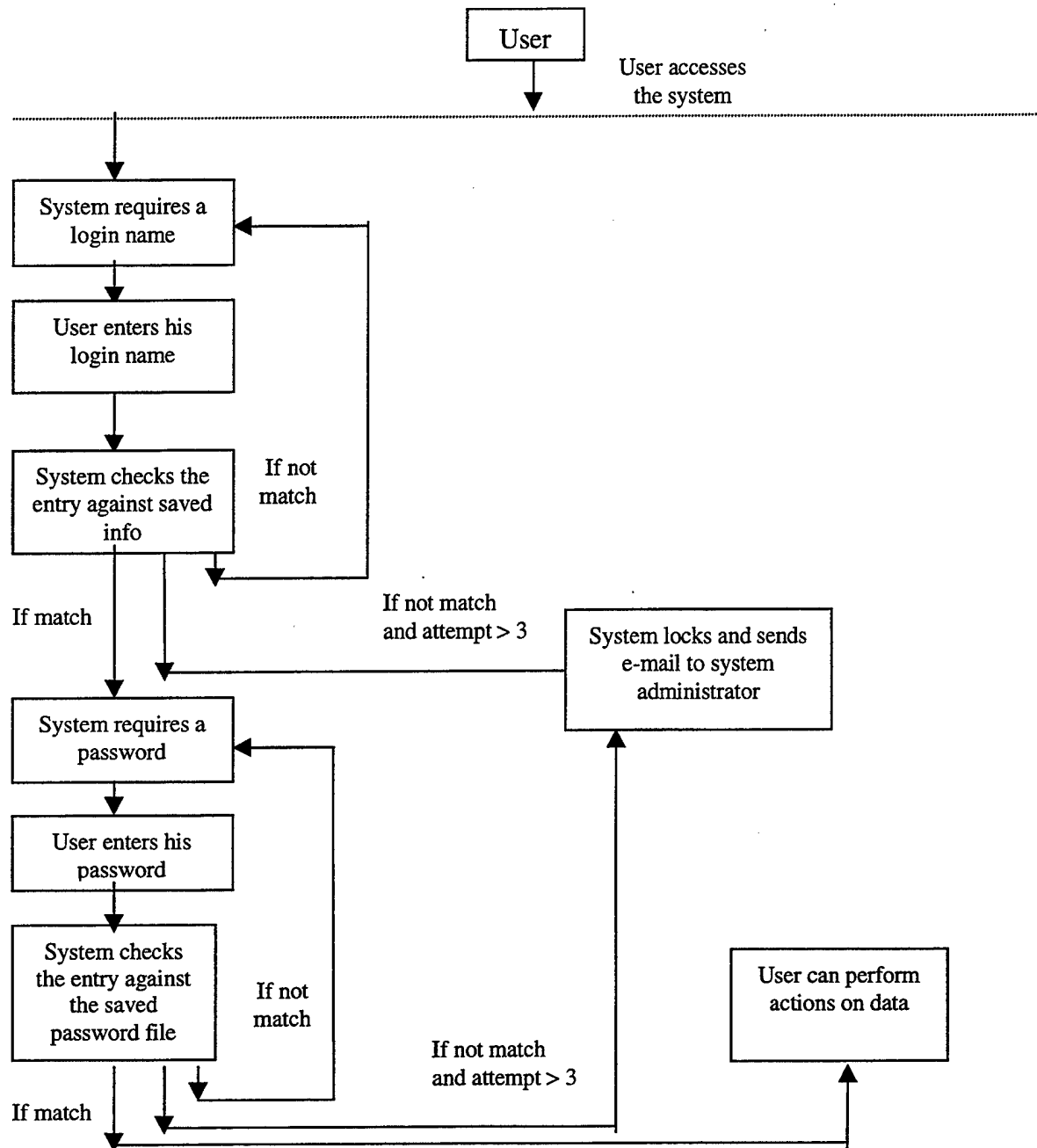


Figure 22. Graphical Representation of Security Level 1 for Project Management Database

VI. ENCRYPTION USING PGP

This chapter provides some basic pieces of information on how Pretty Good Privacy (PGP) works and presents the results of some tests we did using the freeware version 5.0 of PGP for Windows 95. It also contains some discussion on the vulnerabilities of PGP.

A. HOW PGP WORKS

PGP uses IDEA [Ref. 17] for data encryption, RSA [Ref. 17] for key management and digital signatures and MD5 [Ref. 17] as a one-way hash function. PGP's random public keys use a probabilistic primality tester, and get their initial seeds from measuring the keystroke timing and the actual keys struck of the user. It generates random IDEA keys using an algorithm, which is based on the one specified in ANSI X9.17 [Ref. 17] with IDEA instead of DES as the symmetric algorithm. A hashed pass phrase instead of a password is used for encryption of user's private key. The actual operation of PPG consists of five services: authentication, confidentiality, compression, e-mail compatibility, and segmentation. Table 2 presents a brief description for each of the PGP's functions.

Function	Algorithms Used	Description
Message encryption	IDEA, RSA	A message is encrypted using IDEA with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key, and included with the message
Digital signature	RSA, MD5	A hash code of a message is created using MD5. This message digest is encrypted using RSA with the sender's private key, and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP
E-mail compatibility	Radix 64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.
Segmentation		To accommodate maximum message size limitations, PGP performs segmentation and reassembly

Table 2. Summary of PGP Services From Ref [20]

B. PGP EXPERIMENTS

The idea introduced here is to use the PGP for storing data in a database. In this way we can achieve privacy of data critical for the development of a software product and also enforce authentication of people manipulating the data. For being able to say if this idea would work for our model, we did some experiments using the Freeware Version of PGP 5.0 available from M.I.T site [Ref. 22].

For our test purposes we used a potential "personnel" database from the software evolution model. This database consists of 10 records and we assumed that a number of six fields (attributes) are the least required. Also for our tests we used maximum 15 characters for the first name, 20 for the last name and 16 characters for the task. If we represent the unclassified, confidential, secret and top-secret clearances with the codes UN, CF, SC, TS respectively we can use two characters for the field named "Security level." For the social security number we considered a standard length of 11 digits and characters while for the salary we used 4 or 5 digits.

Having in mind that each character or digit needs a byte to be stored, the numbers appeared in Table 3 are expressed in bytes. All the rows with the black numbers are

the plaintext (not encrypted field) of our entries. After each row with the black-colored numbers there is another one with red-colored numbers, which are the respective ciphertext (encrypted field) of the entries after using the PGP. For all the tests we used PGP for encryption and authentication, so from now on when we are talking about encryption this term will include authentication as well.

The two last columns of the table show that encrypting a whole record takes less space than encrypting each field of the record separately. So, for our first record which is 32 bytes in size, a space of 679 bytes is required for encrypting with PGP while a space of 3895 bytes is necessary for storing the same record encrypting each of its fields separately.

We define it as "PGP ratio" the ratio of required space between encryption by field and encryption by record. For a total of 10 records we noticed that the "PGP ratio" is 5.62 meaning that it is required 5.6 times less storage space if we use encryption by record. This ratio becomes considerably bigger when we increase the number of fields in a record from six to eight as shown in Table 4. In this case the PGP ratio is 7.11. For an increase of 86.22% in the actual size

First_name	Last_name	Security Level	Salary	Task	SSN	Address	Phone number	Total	Encryption by field	Encryption by record
2	5	2	4	8	11	30	13	75		721
646	647	646	648	654	654	677	660		5232	
3	5	2	4	8	11	30	13	76		721
650	647	648	648	654	655	677	660		5239	
4	6	2	4	15	11	30	13	85		731
648	652	646	647	661	654	677	660		5245	
4	6	2	5	15	11	30	13	86		733
648	652	647	649	661	655	677	660		5249	
5	7	2	5	15	11	30	13	88		734
647	652	648	647	661	654	677	660		5246	
10	10	2	5	15	11	30	13	96		744
659	659	646	649	661	655	677	660		5266	
11	17	2	5	15	11	30	13	104		752
655	663	647	647	661	655	677	660		5265	
12	18	2	5	15	11	30	13	106		754
661	664	648	649	661	655	677	660		5275	
15	19	2	5	15	11	30	13	110		751
661	665	647	649	661	654	677	660		5274	
15	20	2	5	16	11	30	13	112		761
661	665	647	649	661	654	677	660		5274	
Totals :									52565	7402
PGP Ratio = 7.11										

Table 4. Applying PGP on a Potential "Personnel" Database with Eight Fields

of a plaintext database (no encryption is performed) we have an increase of 44.459% in the size of database if we encrypt by field while there is only an increase of 6.13% if we encrypt by record.

The results from Tables 3 and 4 are showing that it is much better to encrypt the whole record using PGP since it requires less storage space. The large number of fields that might exist in the database is not a problem for encrypting the whole record with PGP, since the more the fields of the record are, the more we save in storage space.

Continuing we describe some other tests we did with PGP. For different sizes of plaintext we measured the respective sizes of ciphertext and the time needed for completing the encryption. Table 5 has different sizes of plaintext records up to 1240000 bytes, which in our opinion is a representative upper bound for data (mostly text type entries) stored in databases of software evolution model. The other two columns give the size of the record after the encryption along with the required for the encryption time. Figure 23 shows a graphical representation of Table 5.

We noticed that the time for the encryption remains considerably low, no matter how big the plaintext size is.

Plaintext	Ciphertext	Time
75	721	1.70
76	721	1.64
85	731	1.67
88	734	1.70
96	744	1.64
106	754	1.62
112	761	1.65
240	802	1.71
400	822	1.84
1000	839	1.83
2000	873	1.8
4000	933	1.79
8000	1053	2.04
32000	1662	2.22
564000	265112	4.91
1240000	1210000	10.33

Table 5. Encrypting Different Sizes of Records with PGP

This is important especially when you are dealing with huge records since it does not take much time to complete an encryption.

It is useful to have a closer look at the ciphertext sizes and see how they vary as the plaintext size is increased. For this reason, we constructed Table 6 and its graph as it is shown on Figure 24 (values up to 8000 bytes or plaintext size) where some more details about ciphertext size variation are appeared.

Assume that an intruder is trying to compromise a file and knows that the file he is looking has a size between 50 and 200 bytes. Even if he has all the ciphertexts, he can

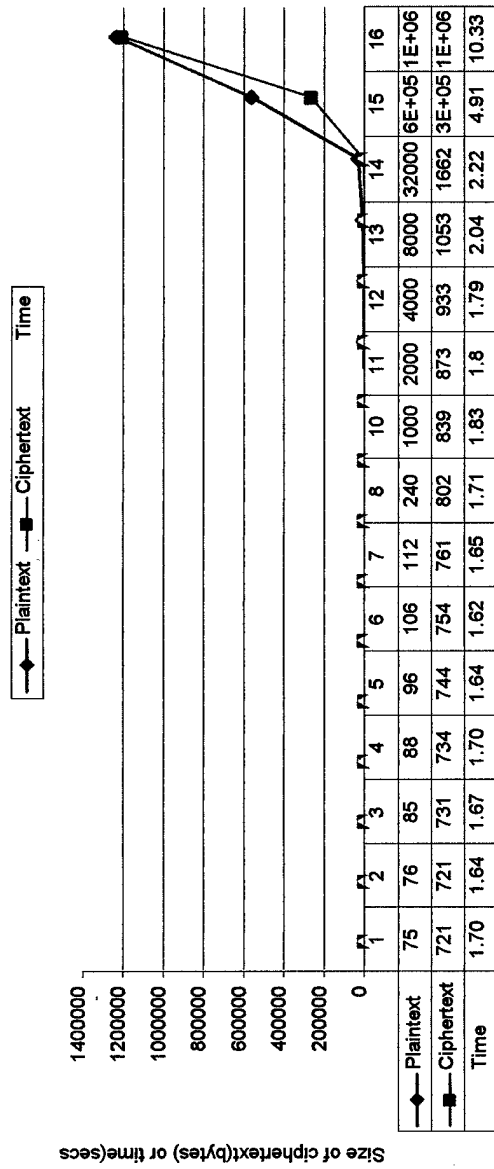


Figure 23. Graphical Representation of Ciphertext Size and Encryption Time for Various Sizes of Records

Plaintext	Ciphertext	Time
75	721	1.70
76	721	1.64
85	731	1.67
86	733	1.65
88	734	1.70
96	744	1.64
104	752	1.68
106	754	1.62
110	751	2.04
112	761	1.65
240	802	1.71
400	822	1.84
1000	839	1.83
2000	873	1.8
4000	933	1.79
8000	1053	2.04

Table 6. Encryption Times (in secs) and Ciphertexts for Various Record Sizes

not say just by monitoring the ciphertext sizes which ones correspond to the range of plaintext sizes he is looking for, since larger file does not mean necessarily larger ciphertext if we are encrypting with PGP.

C. PGP VULNERABILITIES

In this section we discuss some security drawbacks related with PGP.

- a. Compromised pass phrase and secret key. Avoid making your pass phrase a single word. An easy to remember but hard to guess pass phrase can be easily constructed by some creatively nonsensical sayings

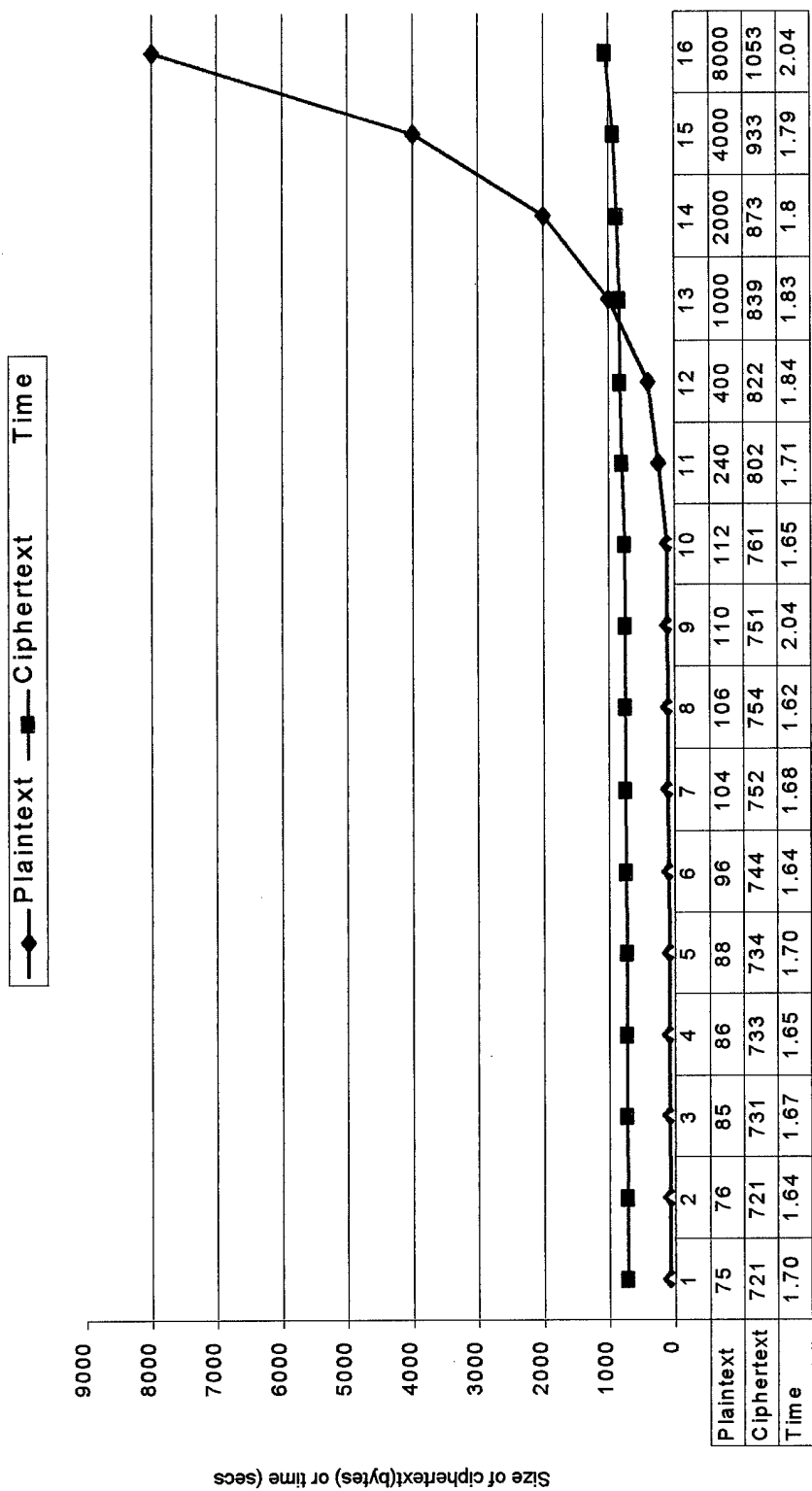


Figure 24. Graphical Representation of Plaintext, Ciphertext and Encryption Time Changes

or very obscure literary quotes. A pass phrase is so much better than a password since it can not be easily guessed.

- b. Public key tampering. When you use someone's public key make sure that it has not been tampered with. If you need to keep a copy of your public and secret key rings in a floppy disk for backup purposes, make sure that it is placed in a safe location.
- c. "Not quite deleted" files. Even if you overwrite the plaintext data on the disk, it may still be possible for a resourceful and determined attacker to recover the data. Faint magnetic traces of the original data remain on the disk after it has been overwritten. You can overwrite the original plaintext file after encryption by using the PGP -w option.
- d. Viruses and Trojan Horses. A specially-tailored hostile computer virus or worm that might infect PGP could be designed to capture your pass phrase or secret key or deciphered messages and covertly write the captured information to a file or send it through a network to the virus's owner. PGP has no defenses against viruses, so you must make sure that his computer environment is virus-free. Also, try to get PGP from a reliable source in order to avoid a Trojan Horse version of PGP, which behaves like PGP

in most respects but does not work the way it's supposed to.

- e. Tempest attacks. This involves a remote detection of the electromagnetic signals from your computer. It could compromise all of your passwords, messages, etc. The technology used for protection against this attack is called "Tempest" and provides a shielding for your computer system.
- f. Exposure on multi-user systems. On such systems there are greater risks of your plaintext or keys or passwords being exposed. PGP cannot protect the data while it is in plaintext form on a compromised system. Nor can it prevent an intruder from using sophisticated measures to read your secret key while it is being used.
- g. Traffic analysis. It is related with an attacker who is trying to infer some useful information by observing where the messages come from and where they are going, the size of the messages and the time of day the messages are sent. PGP alone cannot provide protection against this attack.
- h. Protecting against bogus timestamps. It involves dishonest users creating bogus timestamps on their own public key certificates and signatures. In other words altering the date and time setting on his own

system's clock, a dishonest user can generate his own public key certificates and signatures appearing to have been created at a different time. A solution to this attack would be a trustworthy Certifying Authority that could create notarized signatures with a trustworthy timestamp.

- i. Cryptanalysis. PGP is safe to use if your privacy is not going to be violated by a determined and highly resourceful attacker. Both RSA and IDEA algorithms are secure and very difficult to be cracked unless some vast supercomputer resources are used.

VII. CONCLUSION AND RECCOMENDATIONS

In this thesis, security considerations, including requirements and policies, for the software evolution model was examined. Since the model consists of a number of different databases with different needs for data protection, each database was examined separately as a component of a complex software system.

One critical fact about security is that no data security system is perfectly impenetrable. In order to decide which method is the proper one for a system, one has to ask himself if the information he is trying to protect is more valuable to his attacker than the cost of the attack. This should lead him to protecting himself from the cheapest attacks, while not worrying so much about the very expensive attacks.

Based on the functionality of each database, we recommend that different security policies should be applied. Thus, we defined some Control Classes as the means that could be applied to databases for providing data protection against unauthorized activities. Sets of these Control Classes were used for defining "Security Levels" which reflects the security policy that should be considered for each database. The selected security policy should provide adequate protection to the stored data without

making the operation of the database very complicated and inflexible.

In this research we deal with only the protection of data while it is stored in a repository. The envisioned scheme of the software evolution model where the users will be able to send, receive and retrieve data using a wide area network creates the need for addressing the network security issue. Future work could concentrate on the protection of data while in transit over a network. Furthermore, it would be productive if in the future we could select some different known security models that can meet the criteria for Security Levels we defined and see how they work on actual databases.

In addition to the determination of a Security Level for each database, we did some experiments using PGP for encrypting records. The results are useful for further consideration on how to use PGP for protecting databases.

For better documentation on the PGP issue, we believe that more detailed experiments should be done using computers with increased capabilities. Use of the PGP on an actual database in the future would be helpful for extracting more concrete conclusions.

LIST OF REFERENCES

1. Robert P. Cooke, Jr, "Technology Transfer of the Computer-Aided Prototyping System," Thesis, Naval Postgraduate School, September, 1996.
2. W. Gibbs, "Software Chronic Crisis," Scientific American, Sep. 1994, pp. 36-95.
3. Luigi, "Software Evolution Via Rapid Prototyping," IEEE Computer 22, 5 (May 1989), pp. 13-25.
4. Luigi, "A Graph Model for Software Evolution," IEEE Trans. On Software Eng. 16, 8 (Aug. 1990), pp. 917-927.
5. S. Bodr, "A Model and Algorithms for a Software Evolution Control System," Ph.D. Thesis, Computer Science Department, Naval Postgraduate School, Monterey, CA., December 1993.
6. V. Berrins, ed., "Software Merging and Slicing," IEEE Computer Society Press Tutorial, 1995.
7. John McLean, "A General Theory of composition for Trace Sets Closed Under Selective Interleaving Functions," Proceedings of the 1994 IEEE Symposium on Security and Privacy, pp. 79-93, IEEE Press, May 1994.
8. Colin O'Halloran, "A Calculus of Information Flow," Proceedings of the European Symposium on Research in Computer Security, Toulouse, France, 1990.
9. Joseph A. Goguen and Jose Meseguer, "Security Policies and Security Models," Proceedings of the 1982 IEEE Symposium on Research in Security and Privacy, pp. 11-20 IEEE Press April 1982.
10. Joseph A. Goguen and Jose Meseguer, "Unwinding and Inference Control," Proceedings of the Symposium on Security and Privacy, pp. 75-86, IEEE Computer Society, May 1984.
11. Aris Zakynthinos, "On the composition of Security Properties," Ph.D. Thesis, Department of Electrical and Computer Engineering University of Toronto, 1996.
12. Silvana Castano, Mariagrazia Fugini, Giancarlo Mortella and Pierangela Samarati, "Database Security," Addison-Wesley, Publishing, 1995.

13. Synthia Irvine, Classnotes from the "Advanced Topics in Computer Security," course of the Naval Postgraduate School, 1997.
14. Klaus R. Dittrich and Dirk Jonscher, "Current Trends in Database Technology their Impact on Security Concept," Database Security, VIII (A-60), 1994 IFIP.
15. Elisa Berlino, Suchil Johodia and Pierangela Samarati, "Database Security: Research and Practice," Information Systems Vol. 20, No 7, pp. 537-556, 1995.
16. Dorothy E. Denning, Teresa F. Lunt, Rager R. Schell, Mark Heckman, William Shockley, "A Multilevel Relational Data Model," IEEE 1987.
17. Bruce Schneier, "Applied Cryptography," John Wiley & Sons, Inc., 1996.
18. George Pangalos, "Security guidelines for database Systems development," Database Security, VIII (A-60), IFIP 1994.
19. Jeffrey Gene Kaplan, M.D., M.P.S., "Protecting Sensitive Medical Information," Database Security, VI, Status and Prospects (A-21), IFIP, 1993.
20. Williams Stallings, "Network and Internetwork Security. Principles and Practice", Prentice-Hall, Inc. 1995.
21. D.E. Bell and L.J. LaPadula, "Secure computer Systems: Unified Exposition and Multics Interpretation," Technical Report, The Mitre Corp., 1976.
22. M.I.T. Distribution Site for PGP,
<http://web.mit.edu/network/pgp.html>

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218
2. Dudley Knox Library.....2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5121
3. Chairman, Code CS.....1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943-5121
4. Prof. Bert G.M Lundy, Code CS/LN.....1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943-5121
5. Prof.Luqi, Code CS/LQ.....1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943-5121
6. Embassy of Greece.....1
Naval Attache
2228 Massachusetts Ave., NW
Washington, DC 20008
7. Anastasios X.Rambidis.....2
73,Sofokli Venizelou
12131 Peristeri
Athens
Greece